0 = closure({ S = .Prg }) = .Prg, Prg = .'pgm' Code 'end\_pgm'

1: goto(0, Prg) => { S = Prg. }

2: goto(0, 'pgm') => { Prg = 'pgm'.Code 'end\_pgm', Code = .DeclaVar Code, Code = .DeclaVar,

Code = .DeclFunc Code, Code = .DeclFunc,

DeclVar = .RealType 'semicolon', DeclVar = .RealType 'comma' DeclVar,

DeclVar = .RealType 'assign' Expression 'semicolon',

DeclVar = .RealType 'assign' Expression DeclVar,

DeclFunc = .'main' Scope, DeclFunc = .SpecReturnType FuncParamsFormal Scope,

RealType = .TypeAtomic 'IDENTIFIER',

RealType = .TypeAtomic 'IDENTIFIER' 'SQUAREBEG' Expression 'SQUAREEND',

TypeAtomic = .'LDINT', TypeAtomic = .'DLREAL', TypeAtomic =.'DLSTRING',

TypeAtomic = .'DLCHAR', TypeAtomic = .'DLBOOL' , TypeAtomic = .'DLARRAY',

SpecReturnType = .'DLVOID' 'IDENTIFIER' ,

SpecReturnType = .TypeAtomic'IDENTIFIER'

}

3: goto(0, Code) => { Prg = 'pgm' Code.'end\_pgm'}

4: foto(0, 'end\_pgm') => { Prg = pgm' Code 'end\_pgm' .}

5: goto( 2, DeclVar) => { Code = DeclVar.Code, Code = .DeclaVar Code, Code = .DeclaVar,

Code = .DeclFunc Code, Code = .DeclFunc,

DeclVar = .RealType 'semicolon', DeclVar = .RealType 'comma' DeclVar,

DeclVar = .RealType 'assign' Expression 'semicolon',

DeclVar = .RealType 'assign' Expression DeclVar,

DeclFunc = .'main' Scope, DeclFunc = .SpecReturnType FuncParamsFormal Scope,

RealType = .TypeAtomic 'IDENTIFIER',

RealType = .TypeAtomic 'IDENTIFIER' 'SQUAREBEG' Expression 'SQUAREEND',

TypeAtomic = .'LDINT', TypeAtomic = .'DLREAL', TypeAtomic =.'DLSTRING',

TypeAtomic = .'DLCHAR', TypeAtomic = .'DLBOOL' , TypeAtomic = .'DLARRAY',

SpecReturnType = .'DLVOID' 'IDENTIFIER' ,

SpecReturnType = .TypeAtomic'IDENTIFIER',

Code = DeclVar.}

6: goto(2, DeclFunc) => { Code = DeclFunc. Code, Code = .DeclaVar Code, Code = .DeclaVar,

Code = .DeclFunc Code, Code = .DeclFunc,

DeclVar = .RealType 'semicolon', DeclVar = .RealType 'comma' DeclVar,

DeclVar = .RealType 'assign' Expression 'semicolon',

DeclVar = .RealType 'assign' Expression DeclVar,

DeclFunc = .'main' Scope, DeclFunc = .SpecReturnType FuncParamsFormal Scope,

RealType = .TypeAtomic 'IDENTIFIER',

RealType = .TypeAtomic 'IDENTIFIER' 'SQUAREBEG' Expression 'SQUAREEND',

TypeAtomic = .'LDINT', TypeAtomic = .'DLREAL', TypeAtomic =.'DLSTRING',

TypeAtomic = .'DLCHAR', TypeAtomic = .'DLBOOL' , TypeAtomic = .'DLARRAY',

SpecReturnType = .'DLVOID' 'IDENTIFIER' ,

SpecReturnType = .TypeAtomic'IDENTIFIER',

Code = DeclFunc. }

7: goto(2, RealType) => { DeclVar = RealType .'semicolon', DeclVar = RealType. 'comma' DeclVar,

RealType = .TypeAtomic 'IDENTIFIER' 'SQUAREBEG' Expression 'SQUAREEND',

TypeAtomic = .'LDINT', TypeAtomic = .'DLREAL', TypeAtomic =.'DLSTRING',

TypeAtomic = .'DLCHAR', TypeAtomic = .'DLBOOL' , TypeAtomic = .'DLARRAY',

SpecReturnType = .'DLVOID' 'IDENTIFIER' ,

SpecReturnType = .TypeAtomic'IDENTIFIER'

}

8: goto(2, TypeAtomic ) => { RealType = TypeAtomic.'IDENTIFIER',

RealType = TypeAtomic. 'IDENTIFIER' 'SQUAREBEG' Expression 'SQUAREEND',

TypeAtomic = .'LDINT', TypeAtomic = .'DLREAL', TypeAtomic =.'DLSTRING',

TypeAtomic = .'DLCHAR', TypeAtomic = .'DLBOOL' , TypeAtomic = .'DLARRAY',

SpecReturnType = .'DLVOID' 'IDENTIFIER' ,

SpecReturnType = .TypeAtomic'IDENTIFIER'

}

9: goto(2,'main') => { DeclFunc = 'main' .Scope,

Scope = .'KEYBEG' 'KEYEND', Scope = .'KEYBEG' Commands 'KEYEND'

}

10: goto(2, SpecReturnType ) => {

DeclFunc = SpecReturnType .FuncParamsFormal Scope,

SpecReturnType = .'DLVOID' 'IDENTIFIER' ,

SpecReturnType = .TypeAtomic'IDENTIFIER' ,

FuncParamsReal = .'PARENTHBEG' 'PARENTHEND',

FuncParamsReal = .'PARENTHBEG' ExpressionList 'PARENTHEND'

}

11: goto(2,'DLINT' ) => {TypeAtomic = 'DLINT'.}

12: goto(2, 'DLREAL') => {TypeAtomic = 'DLREAL'.}

13: goto(2, 'DLSTRING') => {TypeAtomic ='DLSTRING'.}

14: goto(2, 'DLCHAR') => { TypeAtomic = 'DLCHAR'.}

15: goto(2, 'DLBOOL') => {TypeAtomic = 'DLBOOL'.}

16: goto(2, 'DLARRAY') => {TypeAtomic = 'DLARRAY'.}

17: GOTO(2, 'DLVOID') => { SpecReturnType = 'DLVOID'. 'IDENTIFIER'}

18: goto(5, Code) => { Code = DeclVar Code.}

19: goto(6, Code) => { Code = DeclFunc Code. }

20: goto(7, 'semicolon') => { DeclVar = RealType 'semicolon'.}

21: goto(7, 'comma') => {DeclVar = RealType 'comma'. DeclVar,

DeclVar = .RealType 'semicolon', DeclVar = .RealType 'comma' DeclVar,

DeclVar = .RealType 'assign' Expression 'semicolon',

DeclVar = .RealType 'assign' Expression DeclVar,

DeclFunc = .'main' Scope, DeclFunc = .SpecReturnType FuncParamsFormal Scope,

RealType = .TypeAtomic 'IDENTIFIER',

RealType = .TypeAtomic 'IDENTIFIER' 'SQUAREBEG' Expression 'SQUAREEND',

TypeAtomic = .'LDINT', TypeAtomic = .'DLREAL', TypeAtomic =.'DLSTRING',

TypeAtomic = .'DLCHAR', TypeAtomic = .'DLBOOL' , TypeAtomic = .'DLARRAY',

SpecReturnType = .'DLVOID' 'IDENTIFIER' ,

SpecReturnType = .TypeAtomic'IDENTIFIER'

}

22: goto(8, 'IDENTIFIER') => {RealType = TypeAtomic 'IDENTIFIER'.,

RealType = TypeAtomic 'IDENTIFIER' .'SQUAREBEG' Expression 'SQUAREEND'

}

23: goto(9, Scope) => {DeclFunc = 'main' Scope.}

24: goto(9, 'KEYBEG') => {Scope = 'KEYBEG' .'KEYEND', Scope = 'KEYBEG' .Commands 'KEYEND',

Commands = .DeclVar Commands, Commands = .DeclVar,

Commands = .Assign Commands, Commands = .Assign, Commands = .LoopWhCom Commands,

Commands = .LoopWhCom,Commands = .CondCom Commands, Commands = .CondCom,

Commands = .PrintCom Commands, Commands = .PrintCom,Commands = .ReadCom Commands,

Commands = .ReadCom, Commands = .CallFunc Commands,Commands = .CallFunc, Commands = .'DLRETURN' Expression 'SEMICOLON',

DeclVar = .RealType 'semicolon', DeclVar = .RealType 'comma' DeclVar,

DeclVar = .RealType 'assign' Expression 'semicolon',

DeclVar = .RealType 'assign' Expression DeclVar,

DeclFunc = .'main' Scope, DeclFunc = .SpecReturnType FuncParamsFormal Scope,

RealType = .TypeAtomic 'IDENTIFIER',

RealType = .TypeAtomic 'IDENTIFIER' 'SQUAREBEG' Expression 'SQUAREEND',

TypeAtomic = .'LDINT', TypeAtomic = .'DLREAL', TypeAtomic =.'DLSTRING',

TypeAtomic = .'DLCHAR', TypeAtomic = .'DLBOOL' , TypeAtomic = .'DLARRAY',

SpecReturnType = .'DLVOID' 'IDENTIFIER' ,

SpecReturnType = .TypeAtomic'IDENTIFIER' ,

Assign = .'IDENTIFIER' 'ASSIGN' Expression 'SEMICOLON',

Assign = .'IDENTIFIER' MatrixIndex 'ASSIGN' Expression 'SEMICOLON',

LoopWhCom = .'DLWHILE' 'PARENTHBEG' Expression 'PARENTHEND' Scope,

CondCom = .'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope,

CondCom = .'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope 'DLELSE' Scope,

PrintCom = .'PRINT' 'PARENTHBEG' 'DLSTRING' 'PARENTHEND' 'SEMICOLON',

PrintCom = .'PRINT' 'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON',

ReadCom = .'READ' 'PARENTHBEG' 'DLSTRING' 'PARENTHEND' 'SEMICOLON',

ReadCom = .'READ' 'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON',

CallFunc = .'READ' FuncParamsReal 'SEMICOLON'

}

25: goto(10, FuncParamsFormal ) => {DeclFunc = SpecReturnType FuncParamsFormal .Scope,

Scope = .'KEYBEG' 'KEYEND', Scope = .'KEYBEG' Commands 'KEYEND'

}

26: goto(10, 'PARENTHBEG') => { FuncParamsReal = 'PARENTHBEG' .'PARENTHEND',

FuncParamsReal = 'PARENTHBEG' .ExpressionList 'PARENTHEND',

ExpressionList = .Expression,

ExpressionList = .ExpressionList 'COMMA' Expression,

Expression = .ExprLogicOr,

ExprLogicOr = .ExprLogicOr 'OR' ExprLogicAnd,

ExprLogicOr = .ExprLogicAnd,

ExprLogicAnd = .ExprLogicAnd 'AND' ExprEquals,

ExprLogicAnd = .ExprEquals,

ExprEquals = .ExprEquals 'EQ' ExprAditiv,

ExprEquals = .ExprMult 'DIFF' ExprAditiv ,

ExprEquals = .ExprMult 'SMALLER' ExprAditiv,

ExprEquals = .ExprMult 'SMALLERE' ExprAditiv,

ExprEquals = .ExprMult 'GREATER' ExprAditiv,

ExprEquals = .ExprMult 'GREATERE' ExprAditiv,

ExprEquals = .ExprAditiv,

ExprAditiv = .ExprAditiv 'PLUS' ExprMult,

ExprAditiv = .ExprAditiv 'MINUS' ExprMult,

ExprAditiv = .ExprMult,

ExprMult = .ExprMult 'MULT' ExprExp,

ExprMult = .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

27: goto(17, 'IDENTIFIER') => { SpecReturnType = 'DLVOID' 'IDENTIFIER'.}

28: goto(21, DeclVar) => {DeclVar = RealType 'comma' DeclVar. }

29: goto(22, 'SQUAREBEG' ) => {RealType = TypeAtomic 'IDENTIFIER' 'SQUAREBEG' .Expression 'SQUAREEND',

Expression = .ExprLogicOr,

ExprLogicOr = .ExprLogicOr 'OR' ExprLogicAnd,

ExprLogicOr = .ExprLogicAnd,

ExprLogicAnd = .ExprLogicAnd 'AND' ExprEquals,

ExprLogicAnd = .ExprEquals,

ExprEquals = .ExprEquals 'EQ' ExprAditiv,

ExprEquals = .ExprMult 'DIFF' ExprAditiv ,

ExprEquals = .ExprMult 'SMALLER' ExprAditiv,

ExprEquals = .ExprMult 'SMALLERE' ExprAditiv,

ExprEquals = .ExprMult 'GREATER' ExprAditiv,

ExprEquals = .ExprMult 'GREATERE' ExprAditiv,

ExprEquals = .ExprAditiv,

ExprAditiv = .ExprAditiv 'PLUS' ExprMult,

ExprAditiv = .ExprAditiv 'MINUS' ExprMult,

ExprAditiv = .ExprMult,

ExprMult = .ExprMult 'MULT' ExprExp,

ExprMult = .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

30: goto(24, 'KEYEND') => {Scope = 'KEYBEG' 'KEYEND'.}

31: goto(24, Commands ) => {Scope = 'KEYBEG' Commands .'KEYEND'}

32: goto(24, DeclVar ) => {Commands = DeclVar .Commands, Commands = DeclVar. ,

Commands = .DeclVar Commands, Commands = .DeclVar,

Commands = .Assign Commands, Commands = .Assign, Commands = .LoopWhCom Commands,

Commands = .LoopWhCom,Commands = .CondCom CommandsCommands = .CondCom,

Commands = .PrintCom Commands, Commands = .PrintCom,Commands = .ReadCom Commands,

Commands = .ReadCom,Commands = .CallFunc Commands,Commands = .CallFunc, Commands = .'DLRETURN' Expression 'SEMICOLON',

DeclVar = .RealType 'semicolon', DeclVar = .RealType 'comma' DeclVar,

DeclVar = .RealType 'assign' Expression 'semicolon',

DeclVar = .RealType 'assign' Expression DeclVar,

DeclFunc = .'main' Scope, DeclFunc = .SpecReturnType FuncParamsFormal Scope,

RealType = .TypeAtomic 'IDENTIFIER',

RealType = .TypeAtomic 'IDENTIFIER' 'SQUAREBEG' Expression 'SQUAREEND',

TypeAtomic = .'LDINT', TypeAtomic = .'DLREAL', TypeAtomic =.'DLSTRING',

TypeAtomic = .'DLCHAR', TypeAtomic = .'DLBOOL' , TypeAtomic = .'DLARRAY',

SpecReturnType = .'DLVOID' 'IDENTIFIER' ,

SpecReturnType = .TypeAtomic'IDENTIFIER' ,

Assign = .'IDENTIFIER' 'ASSIGN' Expression 'SEMICOLON',

Assign = .'IDENTIFIER' MatrixIndex 'ASSIGN' Expression 'SEMICOLON',

LoopWhCom = .'DLWHILE' 'PARENTHBEG' Expression 'PARENTHEND' Scope,

CondCom = .'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope,

CondCom = .'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope 'DLELSE' Scope,

PrintCom = .'PRINT' 'PARENTHBEG' 'DLSTRING' 'PARENTHEND' 'SEMICOLON',

PrintCom = .'PRINT' 'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON',

ReadCom = .'READ' 'PARENTHBEG' 'DLSTRING' 'PARENTHEND' 'SEMICOLON',

ReadCom = .'READ' 'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON',

CallFunc = .'READ' FuncParamsReal 'SEMICOLON'

}

33: goto(24, Assign) => {Commands = Assign .Commands, Commands = Assign.,

Commands = .LoopWhCom Commands,

Commands = .DeclVar Commands, Commands = .DeclVar,

Commands = .Assign Commands, Commands = .Assign, Commands = .LoopWhCom Commands,

Commands = .LoopWhCom,Commands = .CondCom Commands, Commands = .CondCom,

Commands = .PrintCom Commands, Commands = .PrintCom,Commands = .ReadCom Commands,

Commands = .ReadCom,Commands = .CallFunc Commands,Commands = .CallFunc, Commands = .'DLRETURN' Expression 'SEMICOLON',

DeclVar = .RealType 'semicolon', DeclVar = .RealType 'comma' DeclVar,

DeclVar = .RealType 'assign' Expression 'semicolon',

DeclVar = .RealType 'assign' Expression DeclVar,

DeclFunc = .'main' Scope, DeclFunc = .SpecReturnType FuncParamsFormal Scope,

RealType = .TypeAtomic 'IDENTIFIER',

RealType = .TypeAtomic 'IDENTIFIER' 'SQUAREBEG' Expression 'SQUAREEND',

TypeAtomic = .'LDINT', TypeAtomic = .'DLREAL', TypeAtomic =.'DLSTRING',

TypeAtomic = .'DLCHAR', TypeAtomic = .'DLBOOL' , TypeAtomic = .'DLARRAY',

SpecReturnType = .'DLVOID' 'IDENTIFIER' ,

SpecReturnType = .TypeAtomic'IDENTIFIER' ,

Assign = .'IDENTIFIER' 'ASSIGN' Expression 'SEMICOLON',

Assign = .'IDENTIFIER' MatrixIndex 'ASSIGN' Expression 'SEMICOLON',

LoopWhCom = .'DLWHILE' 'PARENTHBEG' Expression 'PARENTHEND' Scope,

CondCom = .'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope,

CondCom = .'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope 'DLELSE' Scope,

PrintCom = .'PRINT' 'PARENTHBEG' 'DLSTRING' 'PARENTHEND' 'SEMICOLON',

PrintCom = .'PRINT' 'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON',

ReadCom = .'READ' 'PARENTHBEG' 'DLSTRING' 'PARENTHEND' 'SEMICOLON',

ReadCom = .'READ' 'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON',

CallFunc = .'READ' FuncParamsReal 'SEMICOLON'

}

34: goto(24, LoopWhCom) => { Commands = LoopWhCom .Commands,

Commands = .DeclVar Commands, Commands = .DeclVar,

Commands = .Assign Commands, Commands = .Assign, Commands = .LoopWhCom Commands,

Commands = .LoopWhCom,Commands = .CondCom CommandsCommands = .CondCom,

Commands = .PrintCom Commands, Commands = .PrintCom,Commands = .ReadCom Commands,

Commands = .ReadCom,Commands = .CallFunc Commands,Commands = .CallFunc, Commands = .'DLRETURN' Expression 'SEMICOLON',

DeclVar = .RealType 'semicolon', DeclVar = .RealType 'comma' DeclVar,

DeclVar = .RealType 'assign' Expression 'semicolon',

DeclVar = .RealType 'assign' Expression DeclVar,

DeclFunc = .'main' Scope, DeclFunc = .SpecReturnType FuncParamsFormal Scope,

RealType = .TypeAtomic 'IDENTIFIER',

RealType = .TypeAtomic 'IDENTIFIER' 'SQUAREBEG' Expression 'SQUAREEND',

TypeAtomic = .'LDINT', TypeAtomic = .'DLREAL', TypeAtomic =.'DLSTRING',

TypeAtomic = .'DLCHAR', TypeAtomic = .'DLBOOL' , TypeAtomic = .'DLARRAY',

SpecReturnType = .'DLVOID' 'IDENTIFIER' ,

SpecReturnType = .TypeAtomic'IDENTIFIER' ,

Assign = .'IDENTIFIER' 'ASSIGN' Expression 'SEMICOLON',

Assign = .'IDENTIFIER' MatrixIndex 'ASSIGN' Expression 'SEMICOLON',

LoopWhCom = .'DLWHILE' 'PARENTHBEG' Expression 'PARENTHEND' Scope,

CondCom = .'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope,

CondCom = .'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope 'DLELSE' Scope,

PrintCom = .'PRINT' 'PARENTHBEG' 'DLSTRING' 'PARENTHEND' 'SEMICOLON',

PrintCom = .'PRINT' 'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON',

ReadCom = .'READ' 'PARENTHBEG' 'DLSTRING' 'PARENTHEND' 'SEMICOLON',

ReadCom = .'READ' 'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON',

CallFunc = .'READ' FuncParamsReal 'SEMICOLON'

}

35: goto(24, CondCom ) => {Commands = CondCom .Commands, Commands = CondCom.,

Commands = .DeclVar Commands, Commands = .DeclVar,

Commands = .Assign Commands, Commands = .Assign, Commands = .LoopWhCom Commands,

Commands = .LoopWhCom,Commands = .CondCom CommandsCommands = .CondCom,

Commands = .PrintCom Commands, Commands = .PrintCom,Commands = .ReadCom Commands,

Commands = .ReadCom,Commands = .CallFunc Commands,Commands = .CallFunc, Commands = .'DLRETURN' Expression 'SEMICOLON',

DeclVar = .RealType 'semicolon', DeclVar = .RealType 'comma' DeclVar,

DeclVar = .RealType 'assign' Expression 'semicolon',

DeclVar = .RealType 'assign' Expression DeclVar,

DeclFunc = .'main' Scope, DeclFunc = .SpecReturnType FuncParamsFormal Scope,

RealType = .TypeAtomic 'IDENTIFIER',

RealType = .TypeAtomic 'IDENTIFIER' 'SQUAREBEG' Expression 'SQUAREEND',

TypeAtomic = .'LDINT', TypeAtomic = .'DLREAL', TypeAtomic =.'DLSTRING',

TypeAtomic = .'DLCHAR', TypeAtomic = .'DLBOOL' , TypeAtomic = .'DLARRAY',

SpecReturnType = .'DLVOID' 'IDENTIFIER' ,

SpecReturnType = .TypeAtomic'IDENTIFIER' ,

Assign = .'IDENTIFIER' 'ASSIGN' Expression 'SEMICOLON',

Assign = .'IDENTIFIER' MatrixIndex 'ASSIGN' Expression 'SEMICOLON',

LoopWhCom = .'DLWHILE' 'PARENTHBEG' Expression 'PARENTHEND' Scope,

CondCom = .'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope,

CondCom = .'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope 'DLELSE' Scope,

PrintCom = .'PRINT' 'PARENTHBEG' 'DLSTRING' 'PARENTHEND' 'SEMICOLON',

PrintCom = .'PRINT' 'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON',

ReadCom = .'READ' 'PARENTHBEG' 'DLSTRING' 'PARENTHEND' 'SEMICOLON',

ReadCom = .'READ' 'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON',

CallFunc = .'READ' FuncParamsReal 'SEMICOLON'

}

36: goto(24, PrintCom ) => {Commands = PrintCom .Commands, Commands = PrintCom.,

Commands = .DeclVar Commands, Commands = .DeclVar,

Commands = .Assign Commands, Commands = .Assign, Commands = .LoopWhCom Commands,

Commands = .LoopWhCom,Commands = .CondCom CommandsCommands = .CondCom,

Commands = .PrintCom Commands, Commands = .PrintCom,Commands = .ReadCom Commands,

Commands = .ReadCom,Commands = .CallFunc Commands,Commands = .CallFunc, Commands = .'DLRETURN' Expression 'SEMICOLON',

DeclVar = .RealType 'semicolon', DeclVar = .RealType 'comma' DeclVar,

DeclVar = .RealType 'assign' Expression 'semicolon',

DeclVar = .RealType 'assign' Expression DeclVar,

DeclFunc = .'main' Scope, DeclFunc = .SpecReturnType FuncParamsFormal Scope,

RealType = .TypeAtomic 'IDENTIFIER',

RealType = .TypeAtomic 'IDENTIFIER' 'SQUAREBEG' Expression 'SQUAREEND',

TypeAtomic = .'LDINT', TypeAtomic = .'DLREAL', TypeAtomic =.'DLSTRING',

TypeAtomic = .'DLCHAR', TypeAtomic = .'DLBOOL' , TypeAtomic = .'DLARRAY',

SpecReturnType = .'DLVOID' 'IDENTIFIER' ,

SpecReturnType = .TypeAtomic'IDENTIFIER' ,

Assign = .'IDENTIFIER' 'ASSIGN' Expression 'SEMICOLON',

Assign = .'IDENTIFIER' MatrixIndex 'ASSIGN' Expression 'SEMICOLON',

LoopWhCom = .'DLWHILE' 'PARENTHBEG' Expression 'PARENTHEND' Scope,

CondCom = .'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope,

CondCom = .'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope 'DLELSE' Scope,

PrintCom = .'PRINT' 'PARENTHBEG' 'DLSTRING' 'PARENTHEND' 'SEMICOLON',

PrintCom = .'PRINT' 'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON',

ReadCom = .'READ' 'PARENTHBEG' 'DLSTRING' 'PARENTHEND' 'SEMICOLON',

ReadCom = .'READ' 'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON',

CallFunc = .'READ' FuncParamsReal 'SEMICOLON'

}

37: goto(24, ReadCom) => {Commands = ReadCom .Commands, Commands = ReadCom.,

Commands = .DeclVar Commands, Commands = .DeclVar,

Commands = .Assign Commands, Commands = .Assign, Commands = .LoopWhCom Commands,

Commands = .LoopWhCom,Commands = .CondCom CommandsCommands = .CondCom,

Commands = .PrintCom Commands, Commands = .PrintCom,Commands = .ReadCom Commands,

Commands = .ReadCom,Commands = .CallFunc Commands,Commands = .CallFunc, Commands = .'DLRETURN' Expression 'SEMICOLON',

DeclVar = .RealType 'semicolon', DeclVar = .RealType 'comma' DeclVar,

DeclVar = .RealType 'assign' Expression 'semicolon',

DeclVar = .RealType 'assign' Expression DeclVar,

DeclFunc = .'main' Scope, DeclFunc = .SpecReturnType FuncParamsFormal Scope,

RealType = .TypeAtomic 'IDENTIFIER',

RealType = .TypeAtomic 'IDENTIFIER' 'SQUAREBEG' Expression 'SQUAREEND',

TypeAtomic = .'LDINT', TypeAtomic = .'DLREAL', TypeAtomic =.'DLSTRING',

TypeAtomic = .'DLCHAR', TypeAtomic = .'DLBOOL' , TypeAtomic = .'DLARRAY',

SpecReturnType = .'DLVOID' 'IDENTIFIER' ,

SpecReturnType = .TypeAtomic'IDENTIFIER' ,

Assign = .'IDENTIFIER' 'ASSIGN' Expression 'SEMICOLON',

Assign = .'IDENTIFIER' MatrixIndex 'ASSIGN' Expression 'SEMICOLON',

LoopWhCom = .'DLWHILE' 'PARENTHBEG' Expression 'PARENTHEND' Scope,

CondCom = .'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope,

CondCom = .'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope 'DLELSE' Scope,

PrintCom = .'PRINT' 'PARENTHBEG' 'DLSTRING' 'PARENTHEND' 'SEMICOLON',

PrintCom = .'PRINT' 'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON',

ReadCom = .'READ' 'PARENTHBEG' 'DLSTRING' 'PARENTHEND' 'SEMICOLON',

ReadCom = .'READ' 'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON',

CallFunc = .'READ' FuncParamsReal 'SEMICOLON'

}

38: goto(24, CallFunc) => {Commands = CallFunc .Commands,Commands = CallFunc.,

Commands = .DeclVar Commands, Commands = .DeclVar,

Commands = .Assign Commands, Commands = .Assign, Commands = .LoopWhCom Commands,

Commands = .LoopWhCom,Commands = .CondCom CommandsCommands = .CondCom,

Commands = .PrintCom Commands, Commands = .PrintCom,Commands = .ReadCom Commands,

Commands = .ReadCom,Commands = .CallFunc Commands,Commands = .CallFunc, Commands = .'DLRETURN' Expression 'SEMICOLON',

DeclVar = .RealType 'semicolon', DeclVar = .RealType 'comma' DeclVar,

DeclVar = .RealType 'assign' Expression 'semicolon',

DeclVar = .RealType 'assign' Expression DeclVar,

DeclFunc = .'main' Scope, DeclFunc = .SpecReturnType FuncParamsFormal Scope,

RealType = .TypeAtomic 'IDENTIFIER',

RealType = .TypeAtomic 'IDENTIFIER' 'SQUAREBEG' Expression 'SQUAREEND',

TypeAtomic = .'LDINT', TypeAtomic = .'DLREAL', TypeAtomic =.'DLSTRING',

TypeAtomic = .'DLCHAR', TypeAtomic = .'DLBOOL' , TypeAtomic = .'DLARRAY',

SpecReturnType = .'DLVOID' 'IDENTIFIER' ,

SpecReturnType = .TypeAtomic'IDENTIFIER' ,

Assign = .'IDENTIFIER' 'ASSIGN' Expression 'SEMICOLON',

Assign = .'IDENTIFIER' MatrixIndex 'ASSIGN' Expression 'SEMICOLON',

LoopWhCom = .'DLWHILE' 'PARENTHBEG' Expression 'PARENTHEND' Scope,

CondCom = .'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope,

CondCom = .'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope 'DLELSE' Scope,

PrintCom = .'PRINT' 'PARENTHBEG' 'DLSTRING' 'PARENTHEND' 'SEMICOLON',

PrintCom = .'PRINT' 'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON',

ReadCom = .'READ' 'PARENTHBEG' 'DLSTRING' 'PARENTHEND' 'SEMICOLON',

ReadCom = .'READ' 'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON',

CallFunc = .'READ' FuncParamsReal 'SEMICOLON'

}

39: goto(24, 'DLRETURN' ) => {Commands = 'DLRETURN' .Expression 'SEMICOLON',

Expression = .ExprLogicOr,

ExprLogicOr = .ExprLogicOr 'OR' ExprLogicAnd,

ExprLogicOr = .ExprLogicAnd,

ExprLogicAnd = .ExprLogicAnd 'AND' ExprEquals,

ExprLogicAnd = .ExprEquals,

ExprEquals = .ExprEquals 'EQ' ExprAditiv,

ExprEquals = .ExprMult 'DIFF' ExprAditiv ,

ExprEquals = .ExprMult 'SMALLER' ExprAditiv,

ExprEquals = .ExprMult 'SMALLERE' ExprAditiv,

ExprEquals = .ExprMult 'GREATER' ExprAditiv,

ExprEquals = .ExprMult 'GREATERE' ExprAditiv,

ExprEquals = .ExprAditiv,

ExprAditiv = .ExprAditiv 'PLUS' ExprMult,

ExprAditiv = .ExprAditiv 'MINUS' ExprMult,

ExprAditiv = .ExprMult,

ExprMult = .ExprMult 'MULT' ExprExp,

ExprMult = .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

40: goto(24, 'IDENTIFIER') => {Assign = 'IDENTIFIER' .'ASSIGN' Expression 'SEMICOLON',

Assign = 'IDENTIFIER' .MatrixIndex 'ASSIGN' Expression 'SEMICOLON',

MatrixIndex = .'SQUAREBEG' Expression 'SQUAREEND'}

41: goto(24, 'DLWHILE' ) => {LoopWhCom = 'DLWHILE' .'PARENTHBEG' Expression 'PARENTHEND' Scope}

42: goto(24, 'DLIF') => {

CondCom = 'DLIF' .'PARENTHBEG' Expression 'PARENTHEND' Scope,

CondCom = 'DLIF' .'PARENTHBEG' Expression 'PARENTHEND' Scope 'DLELSE' Scope

}

43: goto(24, 'PRINT' ) => {

PrintCom = 'PRINT' .'PARENTHBEG' 'DLSTRING' 'PARENTHEND' 'SEMICOLON',

PrintCom = 'PRINT' .'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON'

}

44: goto(24,'READ') => {

ReadCom = 'READ' .'PARENTHBEG' 'DLSTRING' 'PARENTHEND' 'SEMICOLON',

ReadCom = 'READ' .'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON',

CallFunc = 'READ' .FuncParamsReal 'SEMICOLON',

FuncParamsReal = .'PARENTHBEG' 'PARENTHEND',

FuncParamsReal = .'PARENTHBEG' ExpressionList 'PARENTHEND'

}

45: goto(25, Scope) => {DeclFunc = SpecReturnType FuncParamsFormal Scope.}

46: goto(26, 'PARENTHEND') => {FuncParamsReal = 'PARENTHBEG' 'PARENTHEND'.}

47: goto(26, ExpressionList) => {FuncParamsReal = 'PARENTHBEG' ExpressionList .'PARENTHEND',

ExpressionList = ExpressionList .'COMMA' Expression

}

48: goto(26, Expression) => {ExpressionList = Expression.}

49: goto(26, ExprLogicOr) => {Expression = ExprLogicOr., ExprLogicOr = ExprLogicOr .'OR' ExprLogicAnd}

50: goto(26, ExprLogicAnd) => {ExprLogicOr = ExprLogicAnd., ExprLogicAnd = ExprLogicAnd .'AND' ExprEquals}

51: goto(26, ExprEquals) => {ExprLogicAnd = ExprEquals., ExprEquals = ExprEquals .'EQ' ExprAditiv}

52: goto(26, ExprMult) => {

ExprEquals = ExprMult .'DIFF' ExprAditiv ,

ExprEquals = ExprMult .'SMALLER' ExprAditiv,

ExprEquals = ExprMult .'SMALLERE' ExprAditiv,

ExprEquals = ExprMult .'GREATER' ExprAditiv,

ExprEquals = ExprMult .'GREATERE' ExprAditiv,

ExprAditiv = ExprMult.,

ExprMult = ExprMult .'MULT' ExprExp

}

53: goto(26, ExprAditiv) => { ExprEquals = ExprAditiv.,

ExprAditiv = ExprAditiv .'PLUS' ExprMult,

ExprAditiv = ExprAditiv .'MINUS' ExprMult

}

54: goto(26,ExprExp) => {ExprMult = ExprExp,

ExprExp = ExprExp .'POW' ExprUnitary}

55: goto(26, ExprUnitary) => {ExprExp = ExprUnitary.}

56: goto(26, 'UNARY') => {ExprUnitary = 'UNARY' .ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

57: goto(26, 'NOT') => {ExprUnitary = 'NOT' .ExprUnitary ,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

58: goto(26, 'PARENTHBEG') => {ExprUnitary = 'PARENTHBEG' .ExprLogicOr 'PARENTHEND',

ExprLogicOr = .ExprLogicOr 'OR' ExprLogicAnd,

ExprLogicOr = .ExprLogicAnd,

ExprLogicAnd = .ExprLogicAnd 'AND' ExprEquals,

ExprLogicAnd = .ExprEquals,

ExprEquals = .ExprEquals 'EQ' ExprAditiv,

ExprEquals = .ExprMult 'DIFF' ExprAditiv ,

ExprEquals = .ExprMult 'SMALLER' ExprAditiv,

ExprEquals = .ExprMult 'SMALLERE' ExprAditiv,

ExprEquals = .ExprMult 'GREATER' ExprAditiv,

ExprEquals = .ExprMult 'GREATERE' ExprAditiv,

ExprEquals = .ExprAditiv,

ExprAditiv = .ExprAditiv 'PLUS' ExprMult,

ExprAditiv = .ExprAditiv 'MINUS' ExprMult,

ExprAditiv = .ExprMult,

ExprMult = .ExprMult 'MULT' ExprExp,

ExprMult = .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

59: goto(26, 'IDENTIFIER' ) => {ExprUnitary = 'IDENTIFIER'. ,

ExprUnitary = 'IDENTIFIER' .MatrixIndex,

ExprUnitary = 'IDENTIFIER' .FuncParamsReal,

MatrixIndex = .'SQUAREBEG' Expression 'SQUAREEND',

FuncParamsReal = .'PARENTHBEG' 'PARENTHEND',

FuncParamsReal = .'PARENTHBEG' ExpressionList 'PARENTHEND'

}

60: goto(26, Literal) => {ExprUnitary = Literal.}

61: goto(26, 'PARENTHEND') => { FuncParamsReal = 'PARENTHBEG' 'PARENTHEND'.}

62: goto(26, ExpressionList ) => {FuncParamsReal = 'PARENTHBEG' ExpressionList .'PARENTHEND',

ExpressionList = ExpressionList '.COMMA' Expression

}

63: goto(26, Expression) => {ExpressionList = Expression.}

64: goto(29, Expression ) => {RealType = TypeAtomic 'IDENTIFIER' 'SQUAREBEG' Expression .'SQUAREEND'}

65: goto(29, Literal) => {ExprUnitary = Literal.}

66: goto(31, 'KEYEND') => {Scope = 'KEYBEG' Commands 'KEYEND'.}

67: goto(32, Commands) => {Commands = DeclVar Commands.}

68: goto(33, Commands) => {Commands = Assign Commands.}

69: goto(34, Commands) => {Commands = LoopWhCom Commands.}

70: goto(35, Commands) => {Commands = CondCom Commands.}

71: goto(36, Commands) => {Commands = PrintCom Commands.}

72: goto(37, Commands) => {Commands = ReadCom Commands.}

73: goto(38, Commands) => {Commands = CallFunc Commands.}

74: goto(39, Expression) => {Commands = 'DLRETURN' Expression .'SEMICOLON'}

75: goto(40,'ASSIGN') => {Assign = 'IDENTIFIER' 'ASSIGN' .Expression 'SEMICOLON',

Expression = .ExprLogicOr,

ExprLogicOr = .ExprLogicOr 'OR' ExprLogicAnd,

ExprLogicOr = .ExprLogicAnd,

ExprLogicAnd = .ExprLogicAnd 'AND' ExprEquals,

ExprLogicAnd = .ExprEquals,

ExprEquals = .ExprEquals 'EQ' ExprAditiv,

ExprEquals = .ExprMult 'DIFF' ExprAditiv ,

ExprEquals = .ExprMult 'SMALLER' ExprAditiv,

ExprEquals = .ExprMult 'SMALLERE' ExprAditiv,

ExprEquals = .ExprMult 'GREATER' ExprAditiv,

ExprEquals = .ExprMult 'GREATERE' ExprAditiv,

ExprEquals = .ExprAditiv,

ExprAditiv = .ExprAditiv 'PLUS' ExprMult,

ExprAditiv = .ExprAditiv 'MINUS' ExprMult,

ExprAditiv = .ExprMult,

ExprMult = .ExprMult 'MULT' ExprExp,

ExprMult = .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

76: goto(40, MatrixIndex) => {Assign = 'IDENTIFIER' MatrixIndex .'ASSIGN' Expression 'SEMICOLON'}

77: goto(40, 'SQUAREBEG') => {MatrixIndex = 'SQUAREBEG' .Expression 'SQUAREEND'}

78: goto(41, 'PARENTHBEG' ) => {LoopWhCom = 'DLWHILE' 'PARENTHBEG' .Expression 'PARENTHEND' Scope}

79: goto(42, 'PARENTHBEG' ) => {CondCom = 'DLIF' 'PARENTHBEG' .Expression 'PARENTHEND' Scope, CondCom = 'DLIF' 'PARENTHBEG' .Expression 'PARENTHEND' Scope 'DLELSE' Scope}

80: goto(43, 'PARENTHBEG' ) => {

PrintCom = 'PRINT' 'PARENTHBEG' .'DLSTRING' 'PARENTHEND' 'SEMICOLON',

PrintCom = 'PRINT' 'PARENTHBEG' .'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON'

}

81: goto(44, 'PARENTHBEG') => {ReadCom = 'READ' 'PARENTHBEG' .'DLSTRING' 'PARENTHEND' 'SEMICOLON',

ReadCom = 'READ' 'PARENTHBEG' .'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON',

FuncParamsReal = 'PARENTHBEG' .'PARENTHEND',

FuncParamsReal = 'PARENTHBEG' . ExpressionList 'PARENTHEND'

}

82: goto(44, FuncParamsReal ) => {CallFunc = 'READ' FuncParamsReal .'SEMICOLON'}

83: goto(47, 'PARENTHEND') => {FuncParamsReal = 'PARENTHBEG' ExpressionList 'PARENTHEND'.}

84: goto(47, 'COMMA') => {ExpressionList = ExpressionList 'COMMA' .Expression}

85: goto(49, 'OR' ) => {ExprLogicOr = ExprLogicOr 'OR' .ExprLogicAnd,

ExprLogicAnd = .ExprLogicAnd 'AND' ExprEquals,

ExprLogicAnd = .ExprEquals,

ExprEquals = .ExprEquals 'EQ' ExprAditiv,

ExprEquals = .ExprMult 'DIFF' ExprAditiv ,

ExprEquals = .ExprMult 'SMALLER' ExprAditiv,

ExprEquals = .ExprMult 'SMALLERE' ExprAditiv,

ExprEquals = .ExprMult 'GREATER' ExprAditiv,

ExprEquals = .ExprMult 'GREATERE' ExprAditiv,

ExprEquals = .ExprAditiv,

ExprAditiv = .ExprAditiv 'PLUS' ExprMult,

ExprAditiv = .ExprAditiv 'MINUS' ExprMult,

ExprAditiv = .ExprMult,

ExprMult = .ExprMult 'MULT' ExprExp,

ExprMult = .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

86: goto(50,'AND') => {ExprLogicAnd = ExprLogicAnd 'AND' .ExprEquals,

ExprEquals = .ExprEquals 'EQ' ExprAditiv,

ExprEquals = .ExprMult 'DIFF' ExprAditiv ,

ExprEquals = .ExprMult 'SMALLER' ExprAditiv,

ExprEquals = .ExprMult 'SMALLERE' ExprAditiv,

ExprEquals = .ExprMult 'GREATER' ExprAditiv,

ExprEquals = .ExprMult 'GREATERE' ExprAditiv,

ExprEquals = .ExprAditiv,

ExprAditiv = .ExprAditiv 'PLUS' ExprMult,

ExprAditiv = .ExprAditiv 'MINUS' ExprMult,

ExprAditiv = .ExprMult,

ExprMult = .ExprMult 'MULT' ExprExp,

ExprMult = .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

87: goto(51, 'EQ') => {ExprEquals = ExprEquals 'EQ' .ExprAditiv,

ExprAditiv = .ExprAditiv 'PLUS' ExprMult,

ExprAditiv = .ExprAditiv 'MINUS' ExprMult,

ExprAditiv = .ExprMult,

ExprMult = .ExprMult 'MULT' ExprExp,

ExprMult = .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

88: goto(52, 'DIFF' ) => {ExprEquals = ExprMult 'DIFF' .ExprAditiv,

ExprAditiv = .ExprAditiv 'PLUS' ExprMult,

ExprAditiv = .ExprAditiv 'MINUS' ExprMult,

ExprAditiv = .ExprMult,

ExprMult = .ExprMult 'MULT' ExprExp,

ExprMult = .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

89: goto(52, 'SMALLER') => {ExprEquals = ExprMult 'SMALLER' .ExprAditiv,

ExprAditiv = .ExprAditiv 'PLUS' ExprMult,

ExprAditiv = .ExprAditiv 'MINUS' ExprMult,

ExprAditiv = .ExprMult,

ExprMult = .ExprMult 'MULT' ExprExp,

ExprMult = .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

90: goto(52, 'SMALLERE') => {ExprEquals = ExprMult 'SMALLERE' .ExprAditiv,

ExprAditiv = .ExprAditiv 'PLUS' ExprMult,

ExprAditiv = .ExprAditiv 'MINUS' ExprMult,

ExprAditiv = .ExprMult,

ExprMult = .ExprMult 'MULT' ExprExp,

ExprMult = .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

91: goto(52,'GREATER') => {ExprEquals = ExprMult 'GREATER' .ExprAditiv,

ExprAditiv = .ExprAditiv 'PLUS' ExprMult,

ExprAditiv = .ExprAditiv 'MINUS' ExprMult,

ExprAditiv = .ExprMult,

ExprMult = .ExprMult 'MULT' ExprExp,

ExprMult = .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

92: goto(52, 'GREATERE') => {ExprEquals = ExprMult 'GREATERE' .ExprAditiv,

ExprAditiv = .ExprAditiv 'PLUS' ExprMult,

ExprAditiv = .ExprAditiv 'MINUS' ExprMult,

ExprAditiv = .ExprMult,

ExprMult = .ExprMult 'MULT' ExprExp,

ExprMult = .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

93: goto(52, 'MULT') => {ExprMult = ExprMult 'MULT' .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

94: goto(53, 'PLUS' ) => {ExprAditiv = ExprAditiv 'PLUS' .ExprMult,

ExprMult = .ExprMult 'MULT' ExprExp,

ExprMult = .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

95: goto(53, 'MINUS') => {ExprAditiv = ExprAditiv 'MINUS' .ExprMult,

ExprMult = .ExprMult 'MULT' ExprExp,

ExprMult = .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

96: goto(54, 'POW') => {ExprExp = ExprExp 'POW' .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

97: goto(56, ExprUnitary) => {ExprUnitary = 'UNARY' ExprUnitary.}

98: goto(57, ExprUnitary) => {ExprUnitary = 'NOT' ExprUnitary.}

99: goto(58, ExprLogicOr) => {ExprUnitary = 'PARENTHBEG' ExprLogicOr .'PARENTHEND',

ExprLogicOr = ExprLogicOr .'OR' ExprLogicAnd}

100: goto(59, MatrixIndex) => {ExprUnitary = 'IDENTIFIER' MatrixIndex.}

101: goto(59, FuncParamsReal) => {ExprUnitary = 'IDENTIFIER' FuncParamsReal.}

102: goto(59,'SQUAREBEG') => {MatrixIndex = 'SQUAREBEG' .Expression 'SQUAREEND'}

103: goto(59, 'PARENTHBEG') => {FuncParamsReal = 'PARENTHBEG' .'PARENTHEND',

FuncParamsReal = 'PARENTHBEG' .ExpressionList 'PARENTHEND'

ExpressionList = .Expression,

ExpressionList = .ExpressionList 'COMMA' Expression,

Expression = .ExprLogicOr,

ExprLogicOr = .ExprLogicOr 'OR' ExprLogicAnd,

ExprLogicOr = .ExprLogicAnd,

ExprLogicAnd = .ExprLogicAnd 'AND' ExprEquals,

ExprLogicAnd = .ExprEquals,

ExprEquals = .ExprEquals 'EQ' ExprAditiv,

ExprEquals = .ExprMult 'DIFF' ExprAditiv ,

ExprEquals = .ExprMult 'SMALLER' ExprAditiv,

ExprEquals = .ExprMult 'SMALLERE' ExprAditiv,

ExprEquals = .ExprMult 'GREATER' ExprAditiv,

ExprEquals = .ExprMult 'GREATERE' ExprAditiv,

ExprEquals = .ExprAditiv,

ExprAditiv = .ExprAditiv 'PLUS' ExprMult,

ExprAditiv = .ExprAditiv 'MINUS' ExprMult,

ExprAditiv = .ExprMult,

ExprMult = .ExprMult 'MULT' ExprExp,

ExprMult = .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

104: goto(62, 'PARENTHEND') => {FuncParamsReal = 'PARENTHBEG' ExpressionList 'PARENTHEND'.}

105: goto(62, 'COMMA') => {ExpressionList = ExpressionList 'COMMA' .Expression,

Expression = .ExprLogicOr,

ExprLogicOr = .ExprLogicOr 'OR' ExprLogicAnd,

ExprLogicOr = .ExprLogicAnd,

ExprLogicAnd = .ExprLogicAnd 'AND' ExprEquals,

ExprLogicAnd = .ExprEquals,

ExprEquals = .ExprEquals 'EQ' ExprAditiv,

ExprEquals = .ExprMult 'DIFF' ExprAditiv ,

ExprEquals = .ExprMult 'SMALLER' ExprAditiv,

ExprEquals = .ExprMult 'SMALLERE' ExprAditiv,

ExprEquals = .ExprMult 'GREATER' ExprAditiv,

ExprEquals = .ExprMult 'GREATERE' ExprAditiv,

ExprEquals = .ExprAditiv,

ExprAditiv = .ExprAditiv 'PLUS' ExprMult,

ExprAditiv = .ExprAditiv 'MINUS' ExprMult,

ExprAditiv = .ExprMult,

ExprMult = .ExprMult 'MULT' ExprExp,

ExprMult = .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

106: goto(64, 'SQUAREEND') => {RealType = TypeAtomic 'IDENTIFIER' 'SQUAREBEG' Expression 'SQUAREEND'.}

107: goto(74, 'SEMICOLON') => {Commands = 'DLRETURN' Expression 'SEMICOLON'.}

108: goto(75, Expression ) => {Assign = 'IDENTIFIER' 'ASSIGN' Expression .'SEMICOLON'}

109: goto(76, 'ASSIGN') => {Assign = 'IDENTIFIER' MatrixIndex 'ASSIGN' .Expression 'SEMICOLON',

Expression = .ExprLogicOr,

ExprLogicOr = .ExprLogicOr 'OR' ExprLogicAnd,

ExprLogicOr = .ExprLogicAnd,

ExprLogicAnd = .ExprLogicAnd 'AND' ExprEquals,

ExprLogicAnd = .ExprEquals,

ExprEquals = .ExprEquals 'EQ' ExprAditiv,

ExprEquals = .ExprMult 'DIFF' ExprAditiv ,

ExprEquals = .ExprMult 'SMALLER' ExprAditiv,

ExprEquals = .ExprMult 'SMALLERE' ExprAditiv,

ExprEquals = .ExprMult 'GREATER' ExprAditiv,

ExprEquals = .ExprMult 'GREATERE' ExprAditiv,

ExprEquals = .ExprAditiv,

ExprAditiv = .ExprAditiv 'PLUS' ExprMult,

ExprAditiv = .ExprAditiv 'MINUS' ExprMult,

ExprAditiv = .ExprMult,

ExprMult = .ExprMult 'MULT' ExprExp,

ExprMult = .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

110: goto(77, Expression ) => {MatrixIndex = 'SQUAREBEG' Expression .'SQUAREEND'}

111: goto(78, Expression ) => {LoopWhCom = 'DLWHILE' 'PARENTHBEG' Expression .'PARENTHEND' Scope}

112: goto(79, Expression ) => {CondCom = 'DLIF' 'PARENTHBEG' Expression .'PARENTHEND' Scope, CondCom = 'DLIF' 'PARENTHBEG' Expression .'PARENTHEND' Scope 'DLELSE' Scope}

113: goto(80, 'DLSTRING') => {PrintCom = 'PRINT' 'PARENTHBEG' 'DLSTRING' .'PARENTHEND' 'SEMICOLON',

PrintCom = 'PRINT' 'PARENTHBEG' 'DLSTRING' .'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON'

}

114: goto(81, 'DLSTRING' ) => {ReadCom = 'READ' 'PARENTHBEG' 'DLSTRING' .'PARENTHEND' 'SEMICOLON',

ReadCom = 'READ' 'PARENTHBEG' 'DLSTRING' .'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON'

}

115: goto(81, 'PARENTHEND') => {FuncParamsReal = 'PARENTHBEG' 'PARENTHEND'.}

116: goto(81, ExpressionList ) => {FuncParamsReal = 'PARENTHBEG' ExpressionList .'PARENTHEND'}

117: goto(82, 'SEMICOLON') => {CallFunc = 'READ' FuncParamsReal 'SEMICOLON'.}

118: goto(84, Expression) => {ExpressionList = ExpressionList 'COMMA' Expression.}

119: goto(85, ExprLogicAnd) => {ExprLogicOr = ExprLogicOr 'OR' ExprLogicAnd., ExprLogicAnd = ExprLogicAnd .'AND' ExprEquals}

120: goto(86, ExprEquals) => {ExprLogicAnd = ExprLogicAnd 'AND' ExprEquals.,

ExprEquals = ExprEquals .'EQ' ExprAditiv}

121: goto(87, ExprAditiv) => {ExprEquals = ExprEquals 'EQ' ExprAditiv.,

ExprAditiv = ExprAditiv .'PLUS' ExprMult,

ExprAditiv = ExprAditiv .'MINUS' ExprMult

}

122: goto(88,ExprAditiv) ) => {ExprEquals = ExprMult 'DIFF' ExprAditiv.,

ExprAditiv = ExprAditiv .'PLUS' ExprMult,

ExprAditiv = ExprAditiv .'MINUS' ExprMult}

123: goto(89, ExprAditiv) => {ExprEquals = ExprMult 'SMALLER' ExprAditiv.,

ExprAditiv = ExprAditiv .'PLUS' ExprMult,

ExprAditiv = ExprAditiv .'MINUS' ExprMult

}

124: goto(90, ExprAditiv) => {ExprEquals = ExprMult 'SMALLERE' ExprAditiv.,

ExprAditiv = ExprAditiv .'PLUS' ExprMult,

ExprAditiv = ExprAditiv .'MINUS' ExprMult

}

125: goto(91, ExprAditiv ) => {ExprEquals = ExprMult 'GREATER' ExprAditiv.,

ExprAditiv = ExprAditiv .'PLUS' ExprMult,

ExprAditiv = ExprAditiv .'MINUS' ExprMult

}

126: goto(92, ExprAditiv) => {ExprEquals = ExprMult 'GREATERE' ExprAditiv.,

ExprAditiv = ExprAditiv .'PLUS' ExprMult,

ExprAditiv = ExprAditiv .'MINUS' ExprMult

}

127: goto(93, ExprExp) => {ExprMult = ExprMult 'MULT' ExprExp.,

ExprExp = ExprExp .'POW' ExprUnitary}

128: goto(94, ExprMult ) => {ExprAditiv = ExprAditiv 'PLUS' ExprMult.,

ExprMult = ExprMult .'MULT' ExprExp}

129: goto(94, ExprExp) => {ExprMult = ExprExp.,

ExprExp = ExprExp .'POW' ExprUnitary}

130: goto(95, ExprMult) => {ExprAditiv = ExprAditiv 'MINUS' ExprMult.,

ExprMult = ExprMult .'MULT' ExprExp}

131: goto(96, ExprUnitary) => {ExprExp = ExprExp 'POW' ExprUnitary.}

132: goto(99, 'PARENTHEND') => {ExprUnitary = 'PARENTHBEG' ExprLogicOr 'PARENTHEND'.}

133: goto(99, 'OR' ) => {ExprLogicOr = ExprLogicOr 'OR' .ExprLogicAnd}

134: goto(105, Expression) => {ExpressionList = ExpressionList 'COMMA' Expression.}

135: goto(108, 'SEMICOLON') => {Assign = 'IDENTIFIER' 'ASSIGN' Expression 'SEMICOLON'.}

136: goto(109, Expression ) => {Assign = 'IDENTIFIER' MatrixIndex 'ASSIGN' Expression .'SEMICOLON'}

137: goto(110, 'SQUAREEND') => {MatrixIndex = 'SQUAREBEG' Expression 'SQUAREEND'.}

138: goto(111, 'PARENTHEND' ) => {LoopWhCom = 'DLWHILE' 'PARENTHBEG' Expression 'PARENTHEND' .Scope,

Scope = .'KEYBEG' 'KEYEND',

Scope = .'KEYBEG' Commands 'KEYEND'

}

139: goto(112, 'PARENTHEND') => {CondCom = 'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' .Scope,

CondCom = 'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' .Scope 'DLELSE' Scope,

Scope = .'KEYBEG' 'KEYEND',

Scope = .'KEYBEG' Commands 'KEYEND'

}

140: goto(113, 'PARENTHEND' ) => {ReadCom = 'READ' 'PARENTHBEG' 'DLSTRING' 'PARENTHEND' .'SEMICOLON'}

141: goto(113, 'COMMA' ) => {ReadCom = 'READ' 'PARENTHBEG' 'DLSTRING' 'COMMA' .ExpressionList 'PARENTHEND' 'SEMICOLON',

ExpressionList = .Expression,

ExpressionList = .ExpressionList 'COMMA' Expression,

Expression = .ExprLogicOr,

ExprLogicOr = .ExprLogicOr 'OR' ExprLogicAnd,

ExprLogicOr = .ExprLogicAnd,

ExprLogicAnd = .ExprLogicAnd 'AND' ExprEquals,

ExprLogicAnd = .ExprEquals,

ExprEquals = .ExprEquals 'EQ' ExprAditiv,

ExprEquals = .ExprMult 'DIFF' ExprAditiv ,

ExprEquals = .ExprMult 'SMALLER' ExprAditiv,

ExprEquals = .ExprMult 'SMALLERE' ExprAditiv,

ExprEquals = .ExprMult 'GREATER' ExprAditiv,

ExprEquals = .ExprMult 'GREATERE' ExprAditiv,

ExprEquals = .ExprAditiv,

ExprAditiv = .ExprAditiv 'PLUS' ExprMult,

ExprAditiv = .ExprAditiv 'MINUS' ExprMult,

ExprAditiv = .ExprMult,

ExprMult = .ExprMult 'MULT' ExprExp,

ExprMult = .ExprExp,

ExprExp = .ExprExp 'POW' ExprUnitary,

ExprExp = .ExprUnitary,

ExprUnitary = .'UNARY' ExprUnitary,

ExprUnitary = .'NOT' ExprUnitary,

ExprUnitary = .'PARENTHBEG' ExprLogicOr 'PARENTHEND',

ExprUnitary = .'IDENTIFIER' ,

ExprUnitary = .'IDENTIFIER' MatrixIndex,

ExprUnitary = .'IDENTIFIER' FuncParamsReal,

ExprUnitary = .Literal

}

142: goto(116, 'PARENTHEND') => {FuncParamsReal = 'PARENTHBEG' ExpressionList 'PARENTHEND'.}

143: goto(119, 'AND' ) => {ExprLogicAnd = ExprLogicAnd 'AND' .ExprEquals}

144: goto(120, 'EQ') => {ExprEquals = ExprEquals 'EQ' .ExprAditiv}

145: goto(121, 'PLUS' ) => {ExprAditiv = ExprAditiv 'PLUS' .ExprMult}

146: goto(121, 'MINUS' ) => {ExprAditiv = ExprAditiv 'MINUS' .ExprMult}

147: goto(127, 'POW' ) => {ExprExp = ExprExp 'POW' .ExprUnitary}

148: goto(130, 'MULT' ) => {ExprMult = ExprMult 'MULT' .ExprExp}

149: goto(133, ExprLogicAnd) => {ExprLogicOr = ExprLogicOr 'OR' ExprLogicAnd.}

150: goto(136, 'SEMICOLON') => {Assign = 'IDENTIFIER' MatrixIndex 'ASSIGN' Expression 'SEMICOLON'.}

151: goto(138, Scope) => {LoopWhCom = 'DLWHILE' 'PARENTHBEG' Expression 'PARENTHEND' Scope.}

152: goto(139, Scope) => {CondCom = 'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope.,

CondCom = 'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope .'DLELSE' Scope,

}

153: goto(140, 'SEMICOLON') => {ReadCom = 'READ' 'PARENTHBEG' 'DLSTRING' 'PARENTHEND' 'SEMICOLON'.}

154: goto(141, ExpressionList ) => {ReadCom = 'READ' 'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList .'PARENTHEND' 'SEMICOLON'}

155: goto(143,ExprEquals ) => {ExprLogicAnd = ExprLogicAnd 'AND' ExprEquals.}

156: goto(144, ExprAditiv) => {ExprEquals = ExprEquals 'EQ' ExprAditiv.}

157: goto(145, ExprMult) => {ExprAditiv = ExprAditiv 'PLUS' ExprMult.}

158: goto(146, ExprMult) => {ExprAditiv = ExprAditiv 'MINUS' ExprMult.}

159: goto(147, ExprUnitary) => {ExprExp = ExprExp 'POW' ExprUnitary.}

160: goto(148, ExprExp) => {ExprMult = ExprMult 'MULT' ExprExp.}

161: goto(152, 'DLELSE') => {CondCom = 'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope 'DLELSE' .Scope,

Scope = .'KEYBEG' 'KEYEND',

Scope = .'KEYBEG' Commands 'KEYEND'

}

162: goto(154, 'PARENTHEND') => {ReadCom = 'READ' 'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' .'SEMICOLON'}

163: goto(161, Scope) => {CondCom = 'DLIF' 'PARENTHBEG' Expression 'PARENTHEND' Scope 'DLELSE' Scope.}

164: goto(162, 'SEMICOLON') => {ReadCom = 'READ' 'PARENTHBEG' 'DLSTRING' 'COMMA' ExpressionList 'PARENTHEND' 'SEMICOLON'.}