prog: fn\_decl\_list\* main\_prog

main\_prog : [VAR var\_decl ‘;’] stmt\* TK\_END

stmt: TK\_PRINT lexpr ‘;’

stmt: TK\_INPUT ID ‘;’

stmt: TK\_WHEN ‘(’ lexpr ‘)’ TK\_DO stmt\_block

stmt: TK\_IF ‘(’ lexpr ‘)’ TK\_DO stmt\_block TK\_ELSE stmt\_block

stmt: TK\_UNLESS ‘(’ lexpr ‘)’ TK\_DO stmt\_block

stmt: TK\_WHILE ‘(’ lexpr ‘)’ TK\_DO stmt\_block

stmt: TK\_RETURN lexpr ‘;’

stmt: TK\_UNTIL ‘(’ lexpr ‘)’ TK\_DO stmt\_block

stmt: TK\_LOOP stmt\_block

stmt: TK\_DO stmt\_block TK\_WHILE ‘(’ lexpr ‘)’

stmt: TK\_DO stmt\_block TK\_UNTIL ‘(’ lexpr ‘)’

stmt: TK\_REPEAT TK\_NUM ‘:’ stmt\_block

stmt: TK\_FOR ‘(’ lexpr ‘;’ lexpr ‘;’ lexpr ‘)’ TK\_DO stmt\_block

stmt: TK\_NEXT ‘;’

stmt: TK\_BREAK ‘;’

stmt: ID ‘:=’ lexpr ‘;’

stmt: ID ‘+=’ lexpr ‘;’

stmt: ID ‘-=’ lexpr ‘;’

stmt: ID ‘\*=’ lexpr ‘;’

stmt: ID ‘/=’ lexpr ‘;’

stmt: ID ‘%=’ lexpr ‘;’

stmt: ID ‘++’ ‘;’

stmt: ID ‘--’ ‘;’

stmt: ‘--’ ID ‘;’

stmt: ‘++’ ID ‘;’

fn\_decl\_list: TK\_FUNCTION FID ‘:’ DATATYPE ‘(’ [var\_decl] ‘)’ [ VAR var\_decl ‘;’] stmt\_block

var\_decl: ID ‘:’ DATATYPE [‘,’ ID ‘:’ DATATYPE]\*

stmt\_block: ‘{’ stmt+ ‘}’

stmt\_block: stmt

lexpr: nexpr [[TK\_AND nexpr]\* | [TK\_OR nexpr]\*]

nexpr: TK\_NOT ‘(‘ lexpr ‘)’

nexpr: rexpr

rexpr: simple\_expr [(‘<’|‘==’|‘<=’|‘>’|‘>=’|‘!=’) simple\_expr]

simple\_expr: term [(‘+’|‘-’) term]\*

term: factor [(‘\*’|‘/’|‘%’) factor]\*

factor: TK\_NUM

factor: TK\_BOOL

factor: ID [‘++’| ‘--’]

factor: [‘++’|‘--’] ID

factor: ID

factor: ‘(’ lexpr ‘)’

factor: FID ‘(’ [lexpr [‘,’ lexpr]\*] ‘)’

prog: prog\_fn main\_prog

prog\_fn: fn\_decl\_list prog\_fn

prog\_fn: lambda

main\_prog: VAR var\_decl ; main\_stmt TK\_END

main\_prog: main\_stmt TK\_END

main\_stmt: stmt main\_stmt

main\_stmt: lambda

stmt: TK\_PRINT lexpr ;

stmt: TK\_INPUT ID ;

stmt: TK\_WHEN ( lexpr ) TK\_DO stmt\_block

stmt: TK\_IF ( lexpr ) TK\_DO stmt\_block TK\_ELSE stmt\_block

stmt: TK\_UNLESS ( lexpr ) TK\_DO stmt\_block

stmt: TK\_WHILE ( lexpr ) TK\_DO stmt\_block

stmt: TK\_RETURN lexpr ;

stmt: TK\_UNTIL ( lexpr ) TK\_DO stmt\_block

stmt: TK\_LOOP stmt\_block

stmt: TK\_DO stmt\_block DO\_sig

stmt: TK\_REPEAT TK\_NUM : stmt\_block

stmt: TK\_FOR ( lexpr ; lexpr ; lexpr ) TK\_DO stmt\_block

stmt: TK\_NEXT ;

stmt: TK\_BREAK ;

stmt: ID signo

stmt: -- ID ;

stmt: ++ ID ;

DO\_sig: TK\_WHILE ( lexpr )

DO\_sig: TK\_UNTIL ( lexpr )

signo: := lexpr ;

signo: += lexpr ;

signo: -= lexpr ;

signo: \*= lexpr ;

signo: /= lexpr ;

signo: %= lexpr ;

signo: ++ ;

signo: -- ;

fn\_decl\_list: TK\_FUNCTION FID : DATATYPE ( var\_decl ) fn\_decl\_list\_withVAR stmt\_block

fn\_decl\_list\_withVAR: VAR var\_decl ;

fn\_decl\_list\_withVAR: lambda

var\_decl: ID : DATATYPE cont\_data

cont\_data: , ID : DATATYPE cont\_data

cont\_data: lambda

stmt\_block: { stmt stmt\_mas }

stmt\_block: stmt

stmt\_mas: stmt stmt\_mas

stmt\_mas: lambda

lexpr: nexpr nexpr\_prima

nexpr\_prima: TK\_AND lexpr\_and

nexpr\_prima: TK\_OR lexpr\_or

nexpr\_prima: lambda

lexpr\_and: nexpr nexpr\_primab

nexpr\_primab: TK\_AND lexpr\_and

nexpr\_primab: lambda

lexpr\_or: nexpr nexpr\_primac

nexpr\_primac: TK\_OR lexpr\_or

nexpr\_primac: lambda

nexpr: TK\_NOT ( lexpr )

nexpr: rexpr

rexpr: simple\_expr sig

sig: < simple\_expr

sig: == simple\_expr

sig: <= simple\_expr

sig: > simple\_expr

sig: >= simple\_expr

sig: != simple\_expr

sig: lambda

simple\_expr: term t2

t2: + term t2

t2: - term t2

t2: lambda

term: factor fa

fa: \* factor fa

fa: / factor fa

fa: % factor fa

fa: lambda

factor: TK\_NUM

factor: TK\_BOOL

factor: ID s

factor: ++ ID

factor: -- ID

factor: ( lexpr )

factor: FID ( lexpr f2 )

s: ++

s: --

s: lambda

f2: , lexpr f2

f2: lambda

DATATYPE: num

DATATYPE: bool

TK\_BOOL: true

TK\_BOOL: false

end