Documentation

1. **Original Grammar**

```
Prog -> ClassDecl* ProgBody
ClassDecl ->  class id { VarDecl* FuncDef* } ;
ProgBody -> program FuncBody ; FuncDef*
FuncHead -> Type id ( FParams )
FuncDef -> FuncHead FuncBody ;
FuncBody -> { VarDecl* Statement* }
VarDecl -> Type id ArraySize* ;
Statement -> AssignStat ;
             | if ( Expr ) then StatBlock else StatBlock ;
             | for ( Type id AssignOp Expr ; RelExpr ; AssignStat ) StatBlock ;
             | get ( Variable ) ;
             | put ( Expr ) ;
             | return( Expr ) ;
AssignStat -> Variable AssignOp Expr
StatBlock -> { Statement* } | Statement | epsilon
 Expr -> ArithExpr | RelExpr
RelExpr -> ArithExpr RelOp ArithExpr
ArithExpr -> ArithExpr AddOp Term | Term
Sign -> + | -
Term -> Term MultOp Factor | Factor
Factor -> Variable | Idnest* id ( AParams ) | num | ( ArithExpr ) | not Factor | Sign Factor
Variable -> Idnest* id Indice*
Idnest -> id Indice* .
Indice ->  [ ArithExpr ]
ArraySize -> [ INT ]
Type -> int | float | id
FParams -> Type id ArraySize* FParamsTail* | epsilon
AParams -> Expr AParamsTail* | epsilon
FParamsTail -> , Type id ArraySize*
AParamsTail -> , Expr
AssignOp -> =
RelOp -> == | <> | < | > | <= | >=
AddOp -> + | - | or
MultOp -> * | / | and
```

2. **List-generating Productions, Ambiguities, Left Recursions, and Errors**
   - For List-Generating productions, modify all productions A -> B* into the form:
     A -> BList
     BList -> B BList | epsilon

   - Ambiguities:
     1. Expr -> ArithExpr | RelExpr
        RelExpr -> ArithExpr RelOp ArithExpr
     2. Factor -> Variable | IdnestList id (Aparams) | num | (ArithExpr) | not Factor | Sign Factor
        Variable -> IdnestList id IndiceList

   - Left Recursions in grammar:
     1. ArithExpr -> ArithExpr AddOp Term | Term
     2. Term -> Term MultOp Factor | Factor

   - Errors

1. Overlaps occur between the first and follow sets of the production:
   VarDeclList -> VarDecl VarDeclList | epsilon
   - VarDeclList starts with and is followed by Type or id in two productions
2. Overlaps occur between the first and follow sets of the production:
   IdnestList -> Idnest IdnestList | epsilon
   - IdnestList starts with and is followed by an id

3. Transformed Grammar

Prog -> ClassDeclList ProgBody
ClassDeclList -> ClassDecl ClassDeclList | epsilon
ClassDecl -> class id { ClassMemberDeclList } ;
ClassMemberDeclList -> ClassMemberDecl ClassMemberDeclList | epsilon
ClassMemberDecl -> Type id ClassMemberDecl2
ClassMemberDecl2 -> ArraySizeList ; | (FParams) FuncBody ;
ProgBody -> program FuncBody ; FuncDefList
FuncHead -> Type id (FParams)
FuncDefList -> FuncDef FuncDefList | epsilon
FuncDef -> FuncHead FuncBody ;
FuncBody -> { FuncBodyMemberList }
FuncBodyMemberList -> FuncBodyMember FuncBodyMemberList | epsilon
FuncBodyMember -> int id ArraySizeList ; | float id ArraySizeList ; | id FuncBodyMember2 | Statement2
FuncBodyMember2 -> id ArraySizeList ; | IndiceList FuncBodyMember3
FuncBodyMember3 -> . Variable = Expr ; | = Expr ;
StatementList -> Statement StatementList | epsilon
Statement -> Variable = Expr ; | Statement2
Statement2 -> if (Expr) then StatBlock else StatBlock ; | for (Type id = Expr ; ArithExpr RelOp ArithExpr ; Variable = Expr) StatBlock ; | get (Variable) ; | put (Expr) ; | return (Expr) ;
StatBlock -> { StatementList } | Statement | epsilon
Expr -> ArithExpr Expr2
Expr2 -> RelOp ArithExpr | epsilon
ArithExpr -> Term ArithExpr2
ArithExpr2 -> AddOp Term ArithExpr2 | epsilon
Sign -> + | -
Term -> Factor Term2
Term2 -> MultOp Factor Term2 | epsilon
Factor -> id Factor3 Factor2 | num | (ArithExpr) | not Factor | Sign Factor
Factor3 -> IdnestList | IndiceList
Factor2 -> (AParams) | epsilon
Variable -> id IdnestList
IdnestList -> . id IndiceList IdnestList | IndiceList | epsilon
IndiceList -> Indice IndiceList | epsilon
Indice -> [ArithExpr]
ArraySizeList -> ArraySize ArraySizeList | epsilon
ArraySize -> [INT]
Type -> int | float | id
FParams -> Type id ArraySizeList FParamsTailList | epsilon
AParams -> Expr AParamsTailList | epsilon
FParamsTailList -> FParamsTail FParamsTailList | epsilon
FParamsTail -> , Type id ArraySizeList
AParamsTailList -> AParamsTail AParamsTailList | epsilon
AParamsTail -> , Expr
RelOp -> == | <> | < | > | <= | >=
AddOp -> + | - | or
MultOp -> * | / | and

*Notes:

- num includes INT
- After substitution and factorization, RelOp, VarDecl, Idnest, and AssignOp no longer have productions in the transformed grammar; however, the language described is the same.

4. First and Follow Sets

Non-Terminal Symbol	First Set	Follow Set	Non-Terminal Symbol	First Set	Follow Set
ClassDeclList	epsilon, class	program	ClassDecl	class	class, program
ClassMemberDeclList	epsilon, int, float, id	}	ProgBody	program	$
FuncDefList	epsilon, int, float, id	$	FuncBody	{	;
FuncBodyMemberList	epsilon, int, float, id, if, for, get, put, return	}	FuncBodyMember	int, float, id, if, for, get, put, return	int, float, id, if, for, get, put, return, }
FuncBodyMember2	id, epsilon, [, ., =	int, float, id, if, for, get, put, return, }	FuncBodyMember3	., =	int, float, id, if, for, get, put, return, }
StatementList	epsilon, id, if, for, get, put, return	}	Statement2	if, for, get, put, return	id, if, for, get, put, return, else, ;, int, float, }
StatBlock	{, epsilon, id, if, for, get, put, return	else, ;	Expr2	epsilon, ==, <>, <, >, <=, >=	,,), ;
ArithExpr2	epsilon, +, -, or	],), ==, <>, <, >, <=, >=, ,, ;	Sign	+, -	id, num, INT, (, not, +, -
Term2	epsilon, *, /, and	+, -, or, ;,), ,, >, ==, <, <=, >=, }	Factor	id, num, INT, (, not, +, -	*, /, and, +, -, or, ;,), ,, >, ==, <, <=, >=,]
Factor2	(, epsilon	*, /, and, +, -, or, ;,), ,, >, ==, <, <=, >=,]	Variable	id	), =
IdnestList	., epsilon	), =, ;, ,, (, >, ==, <, <=, >=, +, -, or, *, /, and	IndiceList	epsilon, [	.,), =, ;, ,, (, >, ==, <, <=, >=, +, -, or, *, /, and
Indice	[	.,), =, ;, ,, (, >, ==, <, <=, >=, +, -, or, *, /, and, [	ArraySizeList	epsilon, [	,, ;,)
ArraySize	[	[, ;,), ,	Type	int, float, id	Id
FParams	epsilon, int, float, id	)	AParams	epsilon, id, num, INT, (, not, +, -	)
FParamsTailList	epsilon, ,	)	FParamsTail	,	,
AParamsTailList	epsilon, ,	)	AParamsTail	,	,

RelOp	==, <>, <, >, <=, >=	id, num, INT, (, not, +, -	AddOp	+, -, or	id, num, INT, (, not, +, -
MultOp	*, /, and	id, num, INT, (, not, +, -	ClassMember Decl	int, float, id	closecur, int, float, id
FuncHead	int, float, id	{	Statement	id, if, for, get, put, return	id, if, for, get, put, return, else, ;, }
Prog	epsilon, class	$	FuncDef	int, float, id	int, float, id, $
Term	id, num, INT, (, not, +, -	+, -, or, ;,), ,, >, ==, <, <=, >=, }	ArithExpr	id, num, INT, (, not, +, -	],), ==, <>, <, >, <=, >=, ,, ;
Expr	id, num, INT, (, not, +, -	,,), ;	ClassMember Decl2	(, epsilon, [, semi	closecur, int, float, id