	First Set	Follow Set
program	int, void	$
declaration-list	int, void	$
declaration	int, void	$, int, void
declaration'	(, [, ;	$, int, void
var-declaration	int	$, int, void, (, NUM, ID, ;, if, while, return, }
var-declaration'	[, ;	$, int, void
fun-declaration'	(	$, int, void
params	int, void	)
param-list	int	)
param	int	comma,)
compound-stmt	{	$, int, void, }, else, (, NUM, ID, ;, {, if, while, return
local-declarations	ε, int	(, NUM, ID, ;, if, while, return, }
statement-list	ε, (, NUM, ID, ;, {, if, while, return	}, else, (, NUM, ID, ;, {, if, while, return
statement	(, NUM, ID, ;, {, if, while, return	}, else, (, NUM, ID, ;, {, if, while, return
expression-stmt	(, NUM, ID, ;	}, else, (, NUM, ID, ;, {, if, while, return
selection-stmt	if	}, else, (, NUM, ID, ;, {, if, while, return
iteration-stmt	while	}, else, (, NUM, ID, ;, {, if, while, return
return-stmt	return	}, else, (, NUM, ID, ;, {, if, while, return
expression	(, NUM, ID	;,),], comma
expression'	ε, (, NUM, ID, [, *, /	;,),], comma
expression''	ε, (, NUM, ID, *, /	;,),], comma
var	ε, [	*, /, +, -, ;,),], comma, <=, <, >, >=, ==, !=
simple-expression'	ε, *, /	;,),], comma
relop	<=, <, >, >=, ==, !=	(, NUM, ID
additive-expression	(, NUM, ID	;,),], comma
additive-expression'	ε, *, /	<=, <, >, >=, ==, !=, ;,),], comma
addop	+, -	(, NUM, ID
term	(, NUM, ID	+, -, ;,),], comma, <=, <, >, >=, ==, !=
term'	ε, *, /	+, -, ;,),], comma, <=, <, >, >=, ==, !=
mulop	*, /	(, NUM, ID
factor	(, NUM, ID	*, /, +, -, ;,),], comma, <=, <, >, >=, ==, !=
varcall	ε, [, (	*, /, +, -, ;,),], comma, <=, <, >, >=, ==, !=
call	(	*, /, +, -, ;,),], comma, <=, <, >, >=, ==, !=
args	ε, (, NUM, ID	)

arg-list	(, NUM, ID	)