1 void *program* → **program** {{offset= 0}} **id** {{checkaddgreen(*id*.lex, TYPE_PGM)}} **(** *identifier_list* **)** **;**
declarations subprogram_declarations compound_statement **.**
2.1.1 void *identifier_list* → **id** {{checkaddblue(*id*.lex, TYPE_IDLIST)}} *identifier_list′*
2.2.1 void *identifier_list′* → **,** **id** {{checkaddblue(*id*.lex, TYPE_IDLIST)}} *identifier_list′*
2.2.2 void *identifier_list′* → ϵ
3.1.1 void *declarations* → **var** **id** **:** *type* {{checkaddblue(*id*.lex, *type*.type, offset); offset += *type*.width}} **;** *declarations*
3.2.1 void *declarations* → ϵ
4.1 *type*.type *type* → *standard_type* {{*type*.type = *standard_type*.type; *type*.width = *standard_type*.width}}
4.2 *type*.type *type* → **array** **[** **num** **..** **num** **]** **of** *standard_type*
{{*type*.width = $(num_2 - num_1 + 1)$ * *standard_type*.width}}

type.type	←	*standard_type*.type
TYPE_AINT	if	TYPE_INT
TYPE_AREAL	if	TYPE_REAL
ERR	if	ERR
ERR*	otherwise	

5.1 *standard_type*.type *standard_type* → **integer** {{*standard_type*.type = TYPE_INT; *standard_type*.width = 4}}
5.2 *standard_type*.type *standard_type* → **real** {{*standard_type*.type = TYPE_REAL; *standard_type*.width = 8}}
6.1.1 void *subprogram_declarations* → *subprogram_declaration* **;** *subprogram_declarations*
6.2.1 void *subprogram_declarations* → ϵ
7 void *subprogram_declaration* → *subprogram_head declarations*
subprogram_declarations compound_statement {{endgreenscope();}}
8 void *subprogram_head* → **function** **id** {{checkaddgreen(*id*.lex, TYPE_PLACEHOLDER)}}
arguments **:** *standard_type* {{eye_stack.peek().args = *arguments*.str}} **;**

eye_stack.peek().type	←	*standard_type*.type
TYPE_FINT	if	TYPE_INT
TYPE_FREAL	if	TYPE_REAL
ERR	if	ERR
ERR*	otherwise	

9.1 *arguments*.str *arguments* → **(** *parameter_list* **)** {{*arguments*.str = *parameter_list*.str}}
9.2 *arguments*.str *arguments* → ϵ {{*arguments*.str = “”}}
10.1.1 *parameter_list*.str *parameter_list* → **id** **:** *type* {{checkaddblue(*id*.lex, *type*.type, offset); offset += *type*.width}}
parameter_list′ {{*parameter_list*.str = type2str(*type*.type) . *parameter_list′*.str}}
10.2.1 *parameter_list′*.str *parameter_list′* → **;** **id** **:** *type* {{checkaddblue(*id*.lex, *type*.type, offset); offset += *type*.width}}
parameter_list′ {{*parameter_list′*.str = type2str(*type*.type) . *parameter_list*$_1$′.str}}
10.2.2 *parameter_list′*.str *parameter_list′* → ϵ {{*parameter_list′*.str = “”}}
11 void *compound_statement* → **begin** *optional_statements* **end**
12.1 void *optional_statements* → *statement_list*
12.2 void *optional_statements* → ϵ
13.1.1 void *statement_list* → *statement statement_list′*
13.2.1 void *statement_list′* → **;** *statement statement_list′*
13.2.2 void *statement_list′* → ϵ
14.1.1 void *statement* → *variable* **assignop** *expression*

statement.type	←	*variable*.type	*expression*.type
ERR*	if	Undeclared	
ERR	if	ERR	
ERR	if		ERR
VOID	if	TYPE_INT	TYPE_INT
VOID	if	TYPE_FINT	TYPE_INT
VOID	if	TYPE_REAL	TYPE_REAL
VOID	if	TYPE_FREAL	TYPE_REAL
ERR*	otherwise		

14.2.1 void *statement* → *compound_statement*
14.3.1 void *statement* → **if** *expression* {{check(*expression*.type == TYPE_BOOL)}} **then** *statement statement′*
14.4.1 void *statement′* → **else** *statement*
14.4.2 void *statement′* → ϵ
14.5.1 void *statement* → **while** *expression* {{check(*expression*.type == TYPE_BOOL)}} **do** *statement*
15.1.1 *variable*.type *variable* → **id** {{*variable′*.i = gettype(*id*.lex)}} *variable′* {{*variable*.type = *variable′*.type}}
15.2.1 *variable′*.type *variable′* → **[** *expression* **]**

variable′.type	←	*expression*.type	*variable′*.i
ERR*	if		Undeclared
TYPE_INT	if	TYPE_INT	TYPE_AINT
TYPE_REAL	if	TYPE_INT	TYPE_AREAL
ERR	if	ERR	
ERR	if		ERR
ERR*	if	¬TYPE_INT	
ERR*	if		¬TYPE_AINT *and* ¬TYPE_AREAL

15.2.2 *variable′*.type *variable′* → ϵ {{*variable′*.type = *variable′*.i}}
16.1.1 *expression_list*.str *expression_list* → *expression expression_list′*
{{*expression_list*.str = type2str(*expression*.type) . *expression_list′*.str}}
16.2.1 *expression_list′*.str *expression_list′* → **,** *expression expression_list′*
{{*expression_list′*.str = type2str(*expression*.type) . $expression\_list_1$′.str}}
16.2.2 *expression_list′*.str *expression_list′* → ϵ {{*expression_list′*.str = “”}}
17.1.1 *expression*.type *expression* → *simple_expression* {{*expression′*.i = *simple_expression*.type}}
expression′ {{*expression*.type = *expression′*.type}}
17.2.1 *expression′*.type *expression′* → ϵ {{*expression′*.type = *expression′*.i}}
17.2.2 *expression′*.type *expression′* → **relop** *simple_expression*

expression′.type	←	*simple_expression*.type	*expression′*.i
TYPE_BOOL	if	TYPE_INT	TYPE_INT
TYPE_BOOL	if	TYPE_REAL	TYPE_REAL
ERR	if	ERR	
ERR	if		ERR
ERR*	otherwise		

18.1.1 *simple_expression*.type *simple_expression* → *term* {{*simple_expression′*.i = *term*.type}}
simple_expression′ {{*simple_expression*.type = *simple_expression′*.type}}
18.2.1 *simple_expression*.type *simple_expression* → *sign term*
{{ERR* if *term*.type ∉ {TYPE_REAL,TYPE_INT,ERR} }}
{{*simple_expression′*.i = *term*.type}} *simple_expression′* {{*simple_expression*.type = *simple_expression′*.type}}
18.3.1 *simple_expression′*.type *simple_expression′* → **addop** *term simple_expression′*
{{*simple_expression′*.type = $simple\_expression'_1$.type}}

$simple\_expression'_1$.i	←	*simple_expression′*.i	**addop**.attr	*term*.type
TYPE_INT	if	TYPE_INT	+	TYPE_INT
TYPE_INT	if	TYPE_INT	-	TYPE_INT
TYPE_REAL	if	TYPE_REAL	+	TYPE_REAL
TYPE_REAL	if	TYPE_REAL	-	TYPE_REAL
TYPE_BOOL	if	TYPE_BOOL	*or*	TYPE_BOOL
ERR	if	ERR		
ERR	if			ERR
ERR*	otherwise			

18.3.2 *simple_expression′*.type *simple_expression′* → ϵ {{*simple_expression′*.type = *simple_expression′*.i}}
19.1.1 *term*.type *term* → *factor* {{*term′*.i = *factor*.type}} *term′* {{*term*.type = *term′*.type}}
19.2.1 *term′*.type *term′* → **mulop** *factor term′* {{*term*.type = *term′*.type}}

$term'_1$.i	←	*term′*.i	**mulop**.attr	*factor*.type
TYPE_INT	if	TYPE_INT	*	TYPE_INT
TYPE_REAL	if	TYPE_REAL	*	TYPE_REAL
TYPE_REAL	if	TYPE_REAL	/	TYPE_REAL
TYPE_INT	if	TYPE_INT	*div*	TYPE_INT
TYPE_INT	if	TYPE_INT	*mod*	TYPE_INT
TYPE_BOOL	if	TYPE_BOOL	*and*	TYPE_BOOL
ERR	if	ERR		
ERR	if			ERR
ERR*	otherwise			

19.2.2 *term′*.type *term′* → ϵ {{*term′*.type = *term′*.i}}

20.1.1 $factor$.type $factor \rightarrow$ **id** {{$factor'$.i = gettype(id.lex)}} $factor'$ {{$factor$.type = $factor'$.type }}
20.2.1 $factor'$.type $factor' \rightarrow$ **[** $expression$ **]**

$factor'$.type	$\leftarrow$	$expression$.type	$factor'$.i
ERR*	if		Undeclared
TYPE_INT	if	TYPE_INT	TYPE_AINT
TYPE_REAL	if	TYPE_INT	TYPE_AREAL
ERR	if	ERR	
ERR	if		ERR
ERR*	if	¬TYPE_INT	
ERR*	if		¬TYPE_AINT *and* ¬TYPE_AREAL

20.2.2 $factor'$.type $factor' \rightarrow \epsilon$ {{$factor'$.type = $factor'$.i if *declared and* $\in$ {TYPE_INT, TYPE_REAL} }}
20.3.1 $factor'$.type $factor' \rightarrow$ **(** $expression\_list$ **)**
{{$factor'$.type = funtype_to_scalar($factor'$.i); check($expression\_list$.str == get_args($factor'$.i)) }}

$factor'$.type	$\leftarrow$	$factor'$.i
ERR*	if	Undeclared
ERR*	if	¬TYPE_FINT *and* ¬TYPE_FREAL
TYPE_INT	if	TYPE_FINT
TYPE_REAL	if	TYPE_FREAL

20.4.1 $factor$.type $factor \rightarrow$ **num** {{$factor$.type = **num**.type }}
20.5.1 $factor$.type $factor \rightarrow$ **(** $expression$ **)** {{$factor$.type = $expression$.type }}
20.6.1 $factor$.type $factor \rightarrow$ **not** $factor$ {{$factor$.type = $factor_1$.type *unless* $factor_1$.type $\notin$ {TYPE_BOOL,ERR} }}
21.1 void $sign \rightarrow$ +
21.2 void $sign \rightarrow$ -