

Grammar:

```
1  // List of Colons
2
3  semi_colon_list
4      : ';' semi_colon_list_tail
5
6  semi_colon_list_tail
7      : semi_colon_list
8      | ε
9
10 ---
11
12 // Assignment Operators
13
14 assignment_operator //Code Gen = first. Add more later
15     : '='
16     | MUL_ASSIGN
17     | DIV_ASSIGN
18     | MOD_ASSIGN
19     | ADD_ASSIGN
20     | SUB_ASSIGN
21     | LEFT_ASSIGN
22     | RIGHT_ASSIGN
23     | AND_ASSIGN
24     | XOR_ASSIGN
25     | OR_ASSIGN
26
27 ---
28
29 //Data Types
30
31 typed_ID
32     : type_specifier ID
33
34 type
35     : CHAR
36     | SHORT
37     | INT
38     | LONG
39     | FLOAT
40     | DOUBLE
41
42 type_specifier //TODO Const, Volatile later
43     : type
44     | SIGNED type
45     | UNSIGNED type
46
```

```
47 ---
48
49 //Type List
50
51 type_specifier_list
52   : type_specifier type_specifier_list_tail
53
54 type_specifier_list_tail
55   :  $\epsilon$ 
56   | ',' type_specifier_list
57
58 ---
59
60 //Parameter List
61
62 parameter_specifier_list
63   : type_specifier ID parameter_specifier_tail
64
65 parameter_specifier_list_tail
66   :  $\epsilon$ 
67   | ',' parameter_specifier_list
68
69 ---
70
71 //Program
72
73 program
74   : body EOF
75
76 ---
77
78 //Body
79
80 body_typed_ID
81   : typed_ID_common_prefix body_typed_ID_tail
82
83 body_typed_ID_tail
84   :  $\epsilon$ 
85   : body
86
87 body_direct_declaration
88   : direct_declaration body_direct_declaration_tail
89
90 body_direct_declaration_tail
91   :  $\epsilon$ 
92   : body
93
```

```
94  body
95      : body_typed_ID
96      | body_direct_declaration
97
98  ---
99
100 //Function Declaration & Direct Variable Declaration
101
102 typed_ID_common_prefix
103     : typed_ID typed_ID_tail
104     | VOID '(' function_prefix
105
106 typed_ID_tail:
107     : '(' function_prefix //Functions
108     | '=' CONSTANT //TODO expression
109
110 function_prefix
111     : type_specifier_list ')' semi_colon_list //Prototype
112     | parameter_specifier_list ')' function_tail
113
114 function_tail
115     : semi_colon_list //Prototype
116     | '{' statement_list '}' //Function Statements
117
118 ---
119
120 //Direct Declaration
121
122 direct_declaration
123     : type_specifier ID semi_colon_list
124     | CHAR ID '[' ']' '=' STRING_LITERAL semi_colon_list
125
126 ---
127
128 //Statement and Statement List
129
130 statement
131     : compound_statement //Sub statements to loops, conditional, and code blocks
132     | expression_statement //Assignment, Boolean, Arithmetic Expressions
133     | selection_statement //IF Statements
134     | iteration_statement //Loops
135     | semi_colon_list // End of statement one or more ;
136     | direct_declaration
137
138 statement_list
139     : statement statement_list_tail
140
```

```
141 statement_list_tail
142   : statement_list
143   | ε
144
145
146 compound_statement
147   : '{' compound_statement_tail
148
149
150 compound_statement_tail
151   : '}' semi_colon_list
152   | statement_list '}' semi_colon_list
153
154
155 expression_statement
156   : semi_colon_list
157   | expression semi_colon_list
158
159 expression
160   : boolean_expression
161   | assignment_expression
162   | arithmetic_expression
163
164 boolean_expression
165   : //TODO
166
167 arithmetic_expression
168   : //TODO
169
170 assignment_expression
171   : typed_ID assignment_operator assignment_tail semi_colon_list
172
173 assignment_tail
174   : CONSTANT
175   | STRING_LITERAL
176   | boolean_expression
177   | arithmetic_expression
178
179 selection_statement //TODO Case STMT Later
180   | if_stmt
181
182 if_stmt
183   : IF '(' expression ')' statement_list if_stmt_tail
184
185 if_stmt_tail
186   : ε
187   | ELSE statement_list
```

```
188
189 iteration_statement
190     : WHILE '(' expression ')' statement_list
191     | for_stmt
192
193 for_stmt
194     : FOR '(' expression_statement expression_statement for_stmt_tail
195
196 for_stmt_tail
197     : ')' statement_list
198     | expression ')' statement_list
```

Notes - Continue down this route

TODO

- `expr` - arithmetic expressions. Make sure to get precedence (greater precedence last) and associativity correct. From <http://pages.cs.wisc.edu/~fischer/cs536.s08/course.hold/html/NOTES/3.CFG.html#assoc> Remove POW

<code>exp</code>	<code>--> exp PLUS term</code>		<code>exp MINUS term</code>		<code>term</code>
<code>term</code>	<code>--> term TIMES factor</code>		<code>term DIVIDE factor</code>		<code>factor</code>
<code>factor</code>	<code>--> exponent POW factor</code>		<code>exponent</code>		
<code>exponent</code>	<code>--> MINUS exponent</code>		<code>final</code>		
<code>final</code>	<code>--> INTLITERAL</code>		<code>LPAREN exp RPAREN</code>		

- `bexpr` - boolean expressions

```
bexp --> TRUE
bexp --> FALSE
bexp --> bexp OR bexp
bexp --> bexp AND bexp
bexp --> NOT bexp
bexp --> LPAREN bexp RPAREN
```

- `stmt` - add if and while loops to it. Need to make sure no ambiguity in control statements.