Grammar:

```
// List of Colons

semi_colon_list
   :  ';'  semi_colon_list_tail

semi_colon_list_tail
   : semi_colon_list
   | ε

---

// Assignment Operators

assignment_operator //Code Gen = first. Add more later
   : '='
   | MUL_ASSIGN
   | DIV_ASSIGN
   | MOD_ASSIGN
   | ADD_ASSIGN
   | SUB_ASSIGN
   | LEFT_ASSIGN
   | RIGHT_ASSIGN
   | AND_ASSIGN
   | XOR_ASSIGN
   | OR_ASSIGN

---

//Data Types

typed_ID
   : type_specifier ID

type
   : CHAR
   | SHORT
   | INT
   | LONG
   | FLOAT
   | DOUBLE

type_specifier //TODO Const, Volatile later
   : type
   | SIGNED type
   | UNSIGNED type
   | VOID //Compiler will ensure no variable are void while allowing void fcns
```

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

```
94
95  body
96     : body_typed_ID
97     | body_direct_declaration
98
99  ---
100
101 //Function Declaration & Direct Variable Declaration
102
103 typed_ID_common_prefix
104    : typed_ID typed_ID_tail
105
106 typed_ID_tail:
107    : '(' function_prefix //Functions
108    |  assignment_operator expression //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    : statment_list
143    | ε
144
145
146 compound_statement
147    : '{' compound_statement_tail
148
149
150 compound_statement_tail
151    : '}'
152    | statement_list '}'
153
154
155 expression_statement
156    : semi_colon_list
157    | expression
158
159 expression
160    : //TODO
161
162
163 selection_statement //TODO Case STMT Later
164    | if_stmt
165
166 if_stmt
167    : IF '(' expression ')' statement_list if_stmt_tail
168
169 if_stmt_tail
170    : ε
171    | ELSE statement_list
172
173 iteration_statement
174    : WHILE '(' expression ')' statement_list
175    | for_stmt
176
177 for_stmt
178    :  FOR '(' expression_statement expression_statement for_stmt_tail
179
180 for_stmt_tail
181    : ')' statement_list
182    | 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

```
exp      --> exp PLUS term        |  exp MINUS term      |  term
term     --> term TIMES factor    |  term DIVIDE factor  |  factor
factor   --> exponent POW factor  |  exponent
exponent --> MINUS exponent       |  final
final    --> INTLITERAL           |  LPAREN exp RPAREN
```

- 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.