BNF - Elisa Malzoni

```
<function-definition> ::= {<declaration-specifier>}* <declarator>
{<declaration>}* <compound-statement>

<declaration-specifier> ::= <type-specifier>

<type-specifier> ::= void
                   | char
                   | short
                   | int
                   | long
                   | float
                   | double
                   | signed
                   | unsigned

<conditional-expression> ::= <logical-or-expression>
                           | <logical-or-expression> ? <expression> :
<conditional-expression>

<logical-or-expression> ::= <logical-and-expression>
                          | <logical-or-expression> || <logical-and-expression>

<logical-and-expression> ::= <inclusive-or-expression>
                           | <logical-and-expression> &&
<inclusive-or-expression>

<and-expression> ::= <equality-expression>
                   | <and-expression> & <equality-expression>

<equality-expression> ::= <equality-expression> == <relational-expression>
                        | <equality-expression> != <relational-expression>

<additive-expression> ::= <multiplicative-expression>
                        | <additive-expression> + <multiplicative-expression>
                        | <additive-expression> - <multiplicative-expression>

<multiplicative-expression> ::= <cast-expression>
                              | <multiplicative-expression> * <cast-expression>
                              | <multiplicative-expression> / <cast-expression>
                              | <multiplicative-expression> % <cast-expression>

<cast-expression> ::= <unary-expression>
                    | ( <type-name> ) <cast-expression>
```

```
<unary-expression> ::= <postfix-expression>
                     | ++ <unary-expression>
                     | -- <unary-expression>
                     | <unary-operator> <cast-expression>
                     | sizeof <unary-expression>
                     | sizeof <type-name>

<primary-expression> ::= <identifier>
                       | <constant>
                       | <string>
                       | ( <expression> )

<expression> ::= <assignment-expression>
               | <expression> , <assignment-expression>

<assignment-expression> ::= <conditional-expression>
                          | <unary-expression> <assignment-operator>
<assignment-expression>

<assignment-operator> ::= =
                        | *=
                        | /=
                        | %=
                        | +=
                        | -=
                        | &=
                        | ^=
                        | |=

<unary-operator> ::= &
                   | +
                   | -
                   | ~
                   | !

<compound-statement> ::= { {<declaration>}* {<statement>}* }

<statement> ::= <labeled-statement>
              | <expression-statement>
              | <compound-statement>
              | <selection-statement>
              | <iteration-statement>



<expression-statement> ::= {<expression>}? ;
```

```
<selection-statement> ::= se ( <expression> ) <statement>
                        | se ( <expression> ) <statement> senao <statement>

<iteration-statement> ::= enquanto ( <expression> ) <statement>
                        | para ( {<expression>}? ; {<expression>}? ;
{<expression>}? ) <statement>
```