COMPILER CONSTRUCTION (CS402)
C Language Grammar

Notation:

```
======       [ ]     stands for Non terminal
             < >     stands for Terminal/Token coming from lex
```

```
TransitionUnit  → [ExternalDeclaration] [TransitionUnit_a]
TransitionUnit_a    → [ExternalDeclaration] [TransitionUnit_a]
TransitionUnit_a    → ε

ExternalDeclaration → [FunctionDefinition]
ExternalDeclaration → [Declaration]<EndOfStatement>

FunctionDefinition  → [TypeSpecifier] [Declarator]
[FunctionDefinition_a]
FunctionDefinition  → [Declarator][FunctionDefinition_b]

FunctionDefinition_a     → [DeclarationList][CompoundStatement]
FunctionDefinition_a     → [CompoundStatement]

FunctionDefinition_b     → [DeclarationList][CompoundStatement]
FunctionDefinition_b     → [CompoundStatement]

TypeSpecifier       → <void>
TypeSpecifier       → <char>
TypeSpecifier       → <int>
TypeSpecifier       → <float>

Declarator          → [DirectDeclarator]
Declarator          → [Pointer][DirectDeclarator]

DirectDeclarator    → <Identifier> [DirectDeclarator_a]
DirectDeclarator_a  → <OpeningRoundBracket> [DirectDeclarator_b]
DirectDeclarator_a  → <OpeningSquareBracket> [DirectDeclarator_c]
DirectDeclarator_a  → ε
DirectDeclarator_b  →
[Declarator]<ClosingRoundBracket>[DirectDeclarator_a]
DirectDeclarator_b  → [ParameterList]<ClosingRoundBracket>
[DirectDeclarator_a]
DirectDeclarator_b  → <ClosingRoundBracket> [DirectDeclarator_a]
DirectDeclarator_c  → <ClosingSquareBracket>[DirectDeclarator_a]
DirectDeclarator_c  →
<int><ClosingSquareBracket>[DirectDeclarator_a]
```

```
Pointer        → <Multiply>[Pointer_a]
Pointer_a      → [Pointer]
Pointer_a      → ε

DeclarationList     → [Declaration] <EndOfStatement>
[DeclarationList_a]
DeclarationList_a   → [Declaration] <EndOfStatement>
[DeclarationList_a ]
DeclarationList_a   → ε

Declaration         → [TypeSpecifier][Declaration_a]
Declaration_a       → [InitDeclaratorList]
Declaration_a       → ε

InitDeclaratorList  → [InitDeclarator] [InitDeclaratorList_a]
InitDeclaratorList_a     → <Comma>[InitDeclarator]
[InitDeclaratorList_a ]
InitDeclaratorList_a     → ε

InitDeclarator      → [Declarator][InitDeclarator_a]
InitDeclarator_a    → <Assign>[Initializer]
InitDeclarator_a    → ε

Initializer         → [Constant]
Initializer         → <OpeningBraces>[InitializerList]
[Initializer_a]
Initializer_a       → <ClosingBraces>
Initializer_a       → <Comma><ClosingBraces>

InitializerList          → [Initializer][InitializerList_a]
InitializerList_a   →<Comma>[Initializer][InitializerList_a]
InitializerList_a   → ε

ParameterList       → [ParameterDeclaration] [ParameterList_a]
ParameterList_a     → <Comma>[ParameterDeclaration]
[ParameterList_a ]
ParameterList_a     → ε

ParameterDeclaration     → [TypeSpecifier][ParameterDeclaration_a]
ParameterDeclaration_a   → [Declarator]
ParameterDeclaration_a   → ε
```

```
CompoundStatement    → <OpeningBraces>[CompoundStatement_a]
CompoundStatement_a  → <ClosingBraces>
CompoundStatement_a  → [StatementList]<ClosingBraces>
CompoundStatement_a  →  [DeclarationList][CompoundStatement_b]
CompoundStatement_b  →  [StatementList]<ClosingBraces>
CompoundStatement_b  → <ClosingBraces>


StatementList       → [Statement][StatementList_a ]
StatementList_a     → [Statement] [StatementList_a]
StatementList_a           → ε

Statement       → [LabeledStatement]
Statement       → [CompoundStatement]
Statement       → [ExpressionStatement]
Statement       → [SelectionStatement]
Statement       → [IterationStatement]
Statement       → [JumpStatement]

LabeledStatement    → <case>[Constant]<Colon>[Statement]
LabeledStatement    → <default><Colon>[Statement]

ExpressionStatement → <EndOfStatement>
ExpressionStatement → [Expression]<EndOfStatement>


SelectionStatement   →
<if><OpeningRoundBracket>[Expression]<ClosingRoundBracket>
[Statement][SelectionStatement_a]
SelectionStatement_a    → <else>[Statement]
SelectionStatement_a    → ε
SelectionStatement  →
<switch><OpeningRoundBracket>[Expression]<ClosingRoundBracket>[Stat
ement]

JumpStatement  → <continue><EndOfStatement>
JumpStatement  → <break><EndOfStatement>
JumpStatement  → <return> [JumpStatement _a]
JumpStatement_a     → [Expression]<EndOfStatement>

IterationStatement  →
<while><OpeningRoundBracket>[Expression]<ClosingRoundBracket>[State
ment]
```

```
IterationStatement  →
<do>[Statement]<while><OpeningRoundBracket>[Expression]<ClosingRoun
dBracket> <EndOfStatement>

IterationStatement  →
<for><OpeningRoundBracket>[Expression]<EndOfStatement>[Expression]<
EndOfStatement>[Expression]<ClosingRoundBracket>[Statement]

ConditionalExpression    → [EqualityExpression]
[ConditionalExpression_a]
ConditionalExpression_a  →
     <ConditionalOperator>[Expression]<Colon>[ConditionalExpression
     ]
ConditionalExpression_a  → ε
Expression               → [AssignmentExpression][ Expression_a]
Expression_a             → <Comma>[AssignmentExpression]
[Expression_a]
Expression_a             → ε

AssignmentExpression     → [ConditionalExpression]
AssignmentExpression     → [UnaryExpression][AssignmentOperator]
[AssignmentExpression]

EqualityExpression       → [RelationalExpression]
[EqualityExpression_a]
EqualityExpression_a     → <IsEqualTo>[RelationalExpression]
[EqualityExpression_a]
EqualityExpression_a     → <IsNotEqualTo>[RelationalExpression]
[EqualityExpression_a ]
EqualityExpression_ a    → ε

RelationalExpression     → [AdditiveExpression]
[RelationalExpression_a]
RelationalExpression_a   → <LessThan>[AdditiveExpression]
[RelationalExpression_a]
RelationalExpression_a   → <GreaterThan>[AdditiveExpression]
[RelationalExpression_a ]
RelationalExpression_a   → <LessThanEqualTo>[AdditiveExpression]
[RelationalExpression_a]
RelationalExpression_a   →
<GreaterThanEqualTo>[AdditiveExpression] [RelationalExpression_a]
RelationalExpression_a   → ε

AdditiveExpression       → [MultiplicativeExpression]
[AdditiveExpression_a]
```

```
AdditiveExpression_a      → <Add>[MultiplicativeExpression]
[AdditiveExpression_a]
AdditiveExpression_a      → <Subtract>[MultiplicativeExpression]
[AdditiveExpression_a]
AdditiveExpression_a      → ε


MultiplicativeExpression       → [UnaryExpression]
[MultiplicativeExpression_a]
MultiplicativeExpression_a     → <Multiply>[UnaryExpression]
[MultiplicativeExpression_a]
MultiplicativeExpression_a     → <Divide>[UnaryExpression]
[MultiplicativeExpression_a]
MultiplicativeExpression_a     → <Modulus>[UnaryExpression]
[MultiplicativeExpression_a]
MultiplicativeExpression_a     → ε


AssignmentOperator        → <Assign>
AssignmentOperator        → <AddAndAssign>
AssignmentOperator        → <SubtractAndAssign>
AssignmentOperator        → <MultiplyAndAssign>
AssignmentOperator        → <DivideAndAssign>
AssignmentOperator        → <ModulusAndAssign>

Constant             → <Integer>
Constant             → <Character>
Constant             → <Real>

UnaryExpression           → [PostFixExpression]
UnaryExpression           → <Increment>[UnaryExpression]
UnaryExpression           → <Decrement>[UnaryExpression]


PostFixExpression         → [PrimaryExpression][PostFixExpression_a]
PostFixExpression_a       →
<OpeningSquareBracket>[Expression]<ClosingSquareBracket>[PostFixExp
ression_a]

PostFixExpression_a       →
<OpeningRoundBracket>[ PostFixExpression_b]
PostFixExpression_b       →
[ArgumentExpressionList]<ClosingRoundBracket>[PostFixExpression_a]

PostFixExpression_b       →
<ClosingRoundBracket>[PostFixExpression_a]
```

```
PostFixExpression_a      → <Increment>[PostFixExpression_a]
PostFixExpression_a      → <Decrement>[PostFixExpression_a]
PostFixExpression_a      → ε

ArgumentExpressionList   → [AdditiveExpression]
[ArgumentExpressionList_a]
ArgumentExpressionList_a → <Comma>[AdditiveExpression]
[ArgumentExpressionList_a]
ArgumentExpressionList_a → ε

PrimaryExpression        → <Identifier>
PrimaryExpression        → [Constant]
```