C Language Subset Grammar (Complete Version)

Terminals

- **Types**: `int`, `char`
- **Keywords**: `struct`, `if`, `else`, `else if`, `while`, `for`, `return`
- **Pointers**: `*`, `&`
- **Punctuation**: {, }, `;`, `(`, `)`, `=`, `,`, `<`, `>`, `#`, `[`, `]`
- **Operators**:
  - Arithmetic: `+`, `-`, `*`, `/`, `%`, `++`, `--`
  - Assignment: `=`, `+=`, `-=`, `*=`, `/=`, `%=`
  - Relational: `==`, `!=`, `<`, `>`, `<=`, `>=`
  - Logical: `&&`, `||`, `!`
- **Preprocessor**: `#include`
- **Comments**:
  - Line comment: `//`
  - Block comment: `/*`, `*/`

Non-Terminals (Part 1)

$$
\begin{aligned}
Program &\rightarrow (Includes|Comment)\ GlobalDeclarations\ Functions\ MainFunction \\
Includes &\rightarrow (Include|Comment)\ Includes \mid \epsilon \\
Include &\rightarrow \texttt{\#include <}\ HeaderName\ \texttt{>} \mid \texttt{\#include "}\ HeaderName\ \texttt{"} \\
HeaderName &\rightarrow [a-zA-Z0-9.]+ \\
Comment &\rightarrow LineComment \mid BlockComment \\
LineComment &\rightarrow \texttt{//}\ TextUntilNewline \\
BlockComment &\rightarrow \texttt{/*}\ TextUntilEndComment\ \texttt{*/} \\
TextUntilNewline &\rightarrow [{}^{\backslash}n]* \\
TextUntilEndComment &\rightarrow ([{}^{*}]|\texttt{*}+[{}^{*}/])*\texttt{*}* \\
MainFunction &\rightarrow \texttt{int main ( )}\ \{\ Body\ \} \\
Functions &\rightarrow (Function|Comment)\ Functions \mid \epsilon \\
Function &\rightarrow Type\ Identifier\ \texttt{(}\ Parameters\ \texttt{)}\ \{\ Body\ \} \\
Type &\rightarrow \texttt{int} \mid \texttt{char} \mid \texttt{void} \mid \texttt{struct}\ Identifier \\
GlobalDeclarations &\rightarrow (GlobalDecl|Comment)\ GlobalDeclarations \mid \epsilon \\
GlobalDecl &\rightarrow Type\ GlobalVarDecl\ \texttt{;} \\
GlobalVarDecl &\rightarrow Identifier \mid \texttt{*}\ Identifier \mid Identifier\ \texttt{[ ]} \mid Identifier\ \texttt{=}\ Literal \\
Body &\rightarrow LocalDeclarations\ Statements \\
LocalDeclarations &\rightarrow (LocalDecl|Comment)\ LocalDeclarations \mid \epsilon \\
LocalDecl &\rightarrow Type\ VarDecl\ \texttt{;} \\
VarDecl &\rightarrow Identifier \mid \texttt{*}\ Identifier \mid Identifier\ \texttt{[}\ Integer\ \texttt{]} \mid Identifier\ \texttt{=}\ Expression \\
Statements &\rightarrow (Statement|Comment)\ Statements \mid \epsilon
\end{aligned}
$$

Non-Terminals (Part 2)

$$\begin{aligned}
\mathit{Statement} &\rightarrow \texttt{if (} \mathit{Expression} \texttt{) } \{\ \mathit{Statements}\ \}\ \mathit{Else} \\
&\mid \texttt{while (} \mathit{Expression} \texttt{) } \{\ \mathit{Statements}\ \} \\
&\mid \texttt{for (} \mathit{ForInit} \texttt{ ; } \mathit{Expression} \texttt{ ; } \mathit{ForUpdate} \texttt{) } \{\ \mathit{Statements}\ \} \\
&\mid \mathit{ExpressionStmt} \\
&\mid \texttt{printf (} \mathit{FormatString} \texttt{ , } \mathit{Arguments} \texttt{) ;} \\
&\mid \texttt{return } \mathit{ReturnExpr} \texttt{ ;} \\
\mathit{ForInit} &\rightarrow \mathit{AssignExpr} \mid \epsilon \\
\mathit{ForUpdate} &\rightarrow \mathit{AssignExpr} \mid \epsilon \\
\mathit{ExpressionStmt} &\rightarrow \mathit{Expression} \texttt{ ;} \\
\mathit{Else} &\rightarrow \texttt{else } \{\ \mathit{Statements}\ \} \mid \texttt{else if (} \mathit{Expression} \texttt{) } \{\ \mathit{Statements}\ \}\ \mathit{Else} \mid \epsilon \\
\mathit{Expression} &\rightarrow \mathit{AssignExpr} \\
\mathit{AssignExpr} &\rightarrow \mathit{LogicalOrExpr} \mid \mathit{UnaryExpr}\ \mathit{AssignOp}\ \mathit{AssignExpr} \\
\mathit{AssignOp} &\rightarrow \texttt{=} \mid \texttt{+=} \mid \texttt{-=} \mid \texttt{*=} \mid \texttt{/=} \mid \texttt{\%=} \\
\mathit{LogicalOrExpr} &\rightarrow \mathit{LogicalAndExpr} \texttt{ || } \mathit{LogicalOrExpr} \mid \mathit{LogicalAndExpr} \\
\mathit{LogicalAndExpr} &\rightarrow \mathit{EqualityExpr} \texttt{ \&\& } \mathit{LogicalAndExpr} \mid \mathit{EqualityExpr} \\
\mathit{EqualityExpr} &\rightarrow \mathit{RelationalExpr}\ \mathit{EqualityOp}\ \mathit{EqualityExpr} \mid \mathit{RelationalExpr} \\
\mathit{EqualityOp} &\rightarrow \texttt{==} \mid \texttt{!=} \\
\mathit{RelationalExpr} &\rightarrow \mathit{AdditiveExpr}\ \mathit{RelationalOp}\ \mathit{RelationalExpr} \mid \mathit{AdditiveExpr} \\
\mathit{RelationalOp} &\rightarrow \texttt{<} \mid \texttt{>} \mid \texttt{<=} \mid \texttt{>=} \\
\mathit{AdditiveExpr} &\rightarrow \mathit{MultiplicativeExpr}\ \mathit{AdditiveOp}\ \mathit{AdditiveExpr} \mid \mathit{MultiplicativeExpr} \\
\mathit{AdditiveOp} &\rightarrow \texttt{+} \mid \texttt{-} \\
\mathit{MultiplicativeExpr} &\rightarrow \mathit{UnaryExpr}\ \mathit{MultiplicativeOp}\ \mathit{MultiplicativeExpr} \mid \mathit{UnaryExpr} \\
\mathit{MultiplicativeOp} &\rightarrow \texttt{*} \mid \texttt{/} \mid \texttt{\%} \\
\mathit{UnaryExpr} &\rightarrow \mathit{PostfixExpr} \mid \mathit{UnaryOp}\ \mathit{UnaryExpr} \\
\mathit{UnaryOp} &\rightarrow \texttt{+} \mid \texttt{-} \mid \texttt{++} \mid \texttt{--} \mid \texttt{!} \mid \texttt{*} \\
\mathit{PostfixExpr} &\rightarrow \mathit{PrimaryExpr} \mid \mathit{PostfixExpr} \texttt{ ++} \mid \mathit{PostfixExpr} \texttt{ --} \\
\mathit{PrimaryExpr} &\rightarrow \mathit{Identifier} \mid \mathit{Literal} \mid \mathit{StringLit} \mid \texttt{(} \mathit{Expression} \texttt{)} \\
\mathit{FunctionCall} &\rightarrow \mathit{Identifier} \texttt{ (} \mathit{Arguments} \texttt{)} \\
\mathit{Arguments} &\rightarrow \mathit{Expression} \texttt{ , } \mathit{Arguments} \mid \mathit{Expression} \mid \epsilon \\
\mathit{ReturnExpr} &\rightarrow \mathit{Expression} \mid \epsilon \\
\mathit{Identifier} &\rightarrow [a\text{-}zA\text{-}Z\_][a\text{-}zA\text{-}Z0\text{-}9\_]* \\
\mathit{Literal} &\rightarrow \mathit{Integer} \mid \mathit{CharLit} \\
\mathit{Integer} &\rightarrow [0\text{-}9]+ \\
\mathit{CharLit} &\rightarrow \text{'}(\mathit{CharEscape} \mid [^{\text{'}}\backslash n])\text{'} \\
\mathit{StringLit} &\rightarrow \text{"}(\mathit{CharEscape} \mid [^{\text{"}}\backslash n]) * \text{"} \\
\mathit{FormatString} &\rightarrow \text{"}(\mathit{CharEscape} \mid \%([0\text{-}9]+)?[dsc] \mid [^{\text{"}}\%\backslash n]) * \text{"} \\
\mathit{CharEscape} &\rightarrow \backslash[ntr\text{'"}]
\end{aligned}$$

Complete Example Program

```
#include <stdio.h>  // Standard I/O header

/*
 * Global variables section
 * Multi-line comment
 */
char* message = "Hello World";
int numbers[5] = {1, 2, 3, 4, 5};

// Function to calculate factorial recursively
int factorial(int n) {
    // Base case
    if (n <= 1) return 1;

    /* Recursive case */
    return n * factorial(n - 1);
}

int main() {
    char c = 'A';  // Character variable
    int x = 5;
    int* ptr = &x;  // Pointer to x

    printf("%s\n", message);  // Print message

    // Calculate and print factorial
    printf("Factorial of %d is %d\n", x, factorial(x));

    /* Print array elements */
    for (int i = 0; i < 5; i++) {
        printf("%d ", numbers[i]);
    }

    return 0;  // Successful execution
}
```