Context-Free Grammar (CFG) and Implementation Using ANTLR in IntelliJ IDEA

Report

CSI 412: Compiler Laboratory(A)

Student Name	Student ID
Fajla Rabby Khan	011151145

Course Teacher: Mir Moynuddin Ahmed Shibly , Lecturer

Department of Computer Science and Engineering
United International University
Dhaka, Bangladesh

December 25, 2023

Contents

1 Introduction to Context-Free Grammar (CFG) 1

2 Installing IntelliJ IDEA 1

3 Adding ANTLR to IntelliJ IDEA 1

4 About the Grammar 1

5 Key Features of the Grammar 1

5.1 Program (prog): . . . 1
5.2 Header Declaration (headerDeclaration): . . . 1
5.3 Variable Declaration (variableDeclaration): . . . 1
5.4 Function Declaration (functionDeclaration): . . . 1
5.5 Control Flow Statements (controlFlowStatement): . . . 2
5.6 Expression Statement (expressionStatement): . . . 2
5.7 If-Else Statement (ifElseStatement): . . . 2
5.8 For Loop (forLoop): . . . 2
5.9 While Loop (whileLoop): . . . 2
5.10 Function Call (functionCall) . . . 2
5.11 Return Statement (returnStatement): . . . 2
5.12 Compound Statement (compoundStatement): . . . 2
5.13 Data Types (dataType): . . . 2
5.14 Parameters (parameters): . . . 2
5.15 Function Parameters (functionParameters): . . . 2
5.16 Expressions (expression): . . . 2
5.17 Variable Name(variableName): . . . 2
5.18 Function Name(functionName): . . . 2
5.19 Logical Operator, Arithmetic Operator, Relational Operator: . . . 3
5.20 ASSIGN, INCREMENT, DECREMENT, COMMA, SEMICOLON, LPAREN, RPAREN, LBRACE, RBRACE: . . . 3
5.21 STRING, IDENTIFIER, NUMBER: . . . 3
5.22 WS: . . . 3

6 Input and Output Examples 3

6.1 Variable Declaration and Initialization: . . . 3
6.2 If-Else Statement: . . . 3
6.3 For Loop: . . . 4
6.4 While Loop: . . . 4
6.5 Function Declaration and Call: . . . 5
6.6 Syntactically Incorrect Input: . . . 5

7 Conclusion 6

8 References 7

1 | Introduction to Context-Free Grammar (CFG)

Context-Free Grammar (CFG) is a formal grammar that describes the syntactic structure of a language. It consists of a set of production rules that define how sentences in the language can be constructed. CFGs are widely used in the specification of programming languages and document structures.

2 | Installing IntelliJ IDEA

IntelliJ IDEA is a popular integrated development environment (IDE) for Java and other programming languages. Follow these steps to install IntelliJ IDEA:

- Visit the official website [1] and download the version compatible with operating system.
- Run the installer and follow the on-screen instructions to install IntelliJ IDEA on your machine.
- Once installed, launch the IDE.

3 | Adding ANTLR to IntelliJ IDEA

ANTLR (ANother Tool for Language Recognition) is a powerful parser generator that can be used to parse structured text or binary files. To add ANTLR to IntelliJ IDEA, follow these steps:

- Open IntelliJ IDEA.
- Go to ”File” - ”Settings” - ”Plugins.”
- Click on ”Marketplace” and search for ”ANTLR v4 grammar plugin.”
- Install the plugin and restart IntelliJ IDEA.

4 | About the Grammar

The Grammar[2] is designed to represent a simplified C-like programming language. It encompasses the syntax for header declarations, variable declarations, function declarations, control flow statements (if-else, for loop, while loop), expressions, and various other constructs commonly found in programming languages. The grammar is defined in ANTLR (ANother Tool for Language Recognition).

5 | Key Features of the Grammar

The grammar defines a simple programming language syntax.

5.1 | Program (prog):

- Represents a program consisting of multiple statements.
- The program can include header declarations, variable declarations, function declarations, control flow statements, expression statements, function calls, and return statements.

5.2 | Header Declaration (headerDeclaration):

- Specifies inclusion of external headers using the `'#include'` directive.

5.3 | Variable Declaration (variableDeclaration):

- Declares variables with optional assignment of expressions.

5.4 | Function Declaration (functionDeclaration):

- Declares functions with a return type, function name, optional parameters, and a compound statement.

5.5 | Control Flow Statements (controlFlowStatement):

- Represents control flow statements such as if-else statements, for loops, and while loops.

5.6 | Expression Statement (expressionStatement):

- Allows the use of expressions followed by a semicolon.

5.7 | If-Else Statement (ifElseStatement):

- Represents if-else statements with optional else part.

5.8 | For Loop (forLoop):

- Specifies the syntax for a for loop, including variable declaration or initialization, loop condition, and iteration.

5.9 | While Loop (whileLoop):

- Represents a while loop with a loop condition.

5.10 | Function Call (functionCall)

- Invokes functions with optional parameters.

5.11 | Return Statement (returnStatement):

- Indicates the return of a value from a function.

5.12 | Compound Statement (compoundStatement):

- Represents a compound statement enclosed in curly braces, containing multiple statements.

5.13 | Data Types (dataType):

- Defines basic data types such as int, float, double, and string.

5.14 | Parameters (parameters):

- Specifies the parameters for function declarations.

5.15 | Function Parameters (functionParameters):

- Represents parameters for function calls.

5.16 | Expressions (expression):

- Defines various types of expressions, including function calls, logical, arithmetic, and relational expressions, variable names, strings, numbers, and parentheses.

5.17 | Variable Name(variableName):

- Represents identifiers for variables.

5.18 | Function Name(functionName):

- Represents identifiers for function names.

5.19 | Logical Operator, Arithmetic Operator, Relational Operator:

- Defines logical (`'&&'`, `'||'`), arithmetic (`'+'`, `'-'`, `'*'`, `'/'`) and relational (`'>'`, `'<'`, `'>='`, `'<='`, `'=='`, `'!='`) operators respectively.

5.20 | ASSIGN, INCREMENT, DECREMENT, COMMA, SEMICOLON, LPAREN, RPAREN, LBRACE, RBRACE:

- Define specific symbols used in the language.

5.21 | STRING, IDENTIFIER, NUMBER:

- Define patterns for strings, identifiers, and numbers, respectively.

5.22 | WS:

- Defines whitespace and instructs the parser to skip it.

The grammar provides a foundation for parsing and understanding the syntax of a programming language. It covers a range of constructs commonly found in imperative languages.

6 | Input and Output Examples

Here are a few examples illustrating the use of the Grammar.

6.1 | Variable Declaration and Initialization:

```
#include<"011151145_project">

int main() {
    int x = 5;
    float y = 10.5;
    return 0;
}
```

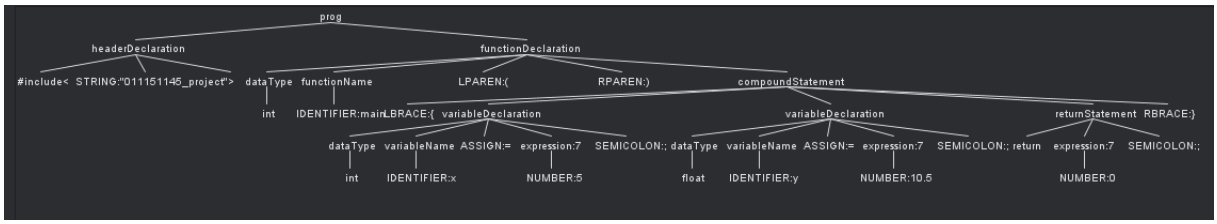

Figure 6.1: Parse tree for Variable Declaration and Initialization (Zoom the PDF)

6.2 | If-Else Statement:

```
int main() {
    int a = 10;
    if (a > 5) {
        printf("Greater than 5\n");
    } else {
        printf("Less than or equal to 5\n");
    }
    return 0;
}
```

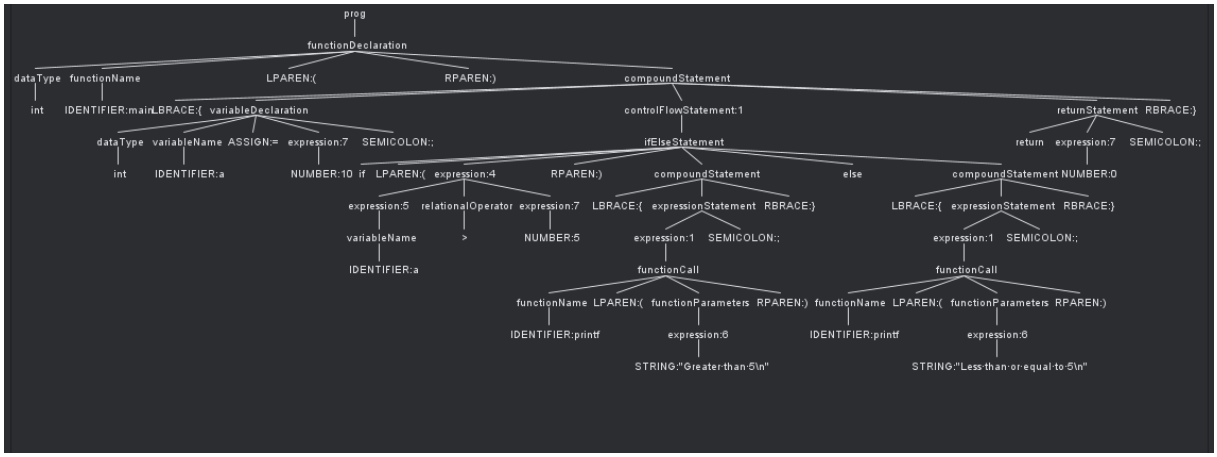

Figure 6.2: Parse tree for If-Else Statement (Zoom the PDF)

6.3 | For Loop:

```
int main() {
    for (int i = 0; i < 5; i++) {
        printf("%d ", i);
    }
    return 0;
}
```

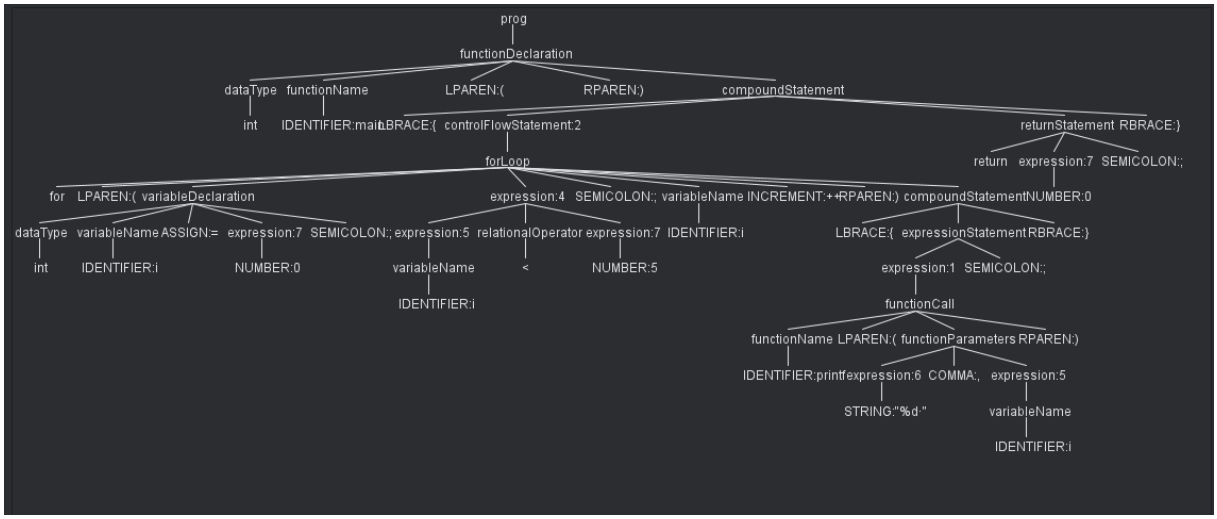

Figure 6.3: Parse tree for For Loop (Zoom the PDF)

6.4 | While Loop:

```
int main() {
    int a = 3;
    while (a > 0) {
        a--;
        printf("Countdown: %d\n", a);
    }
    return 0;
}
```

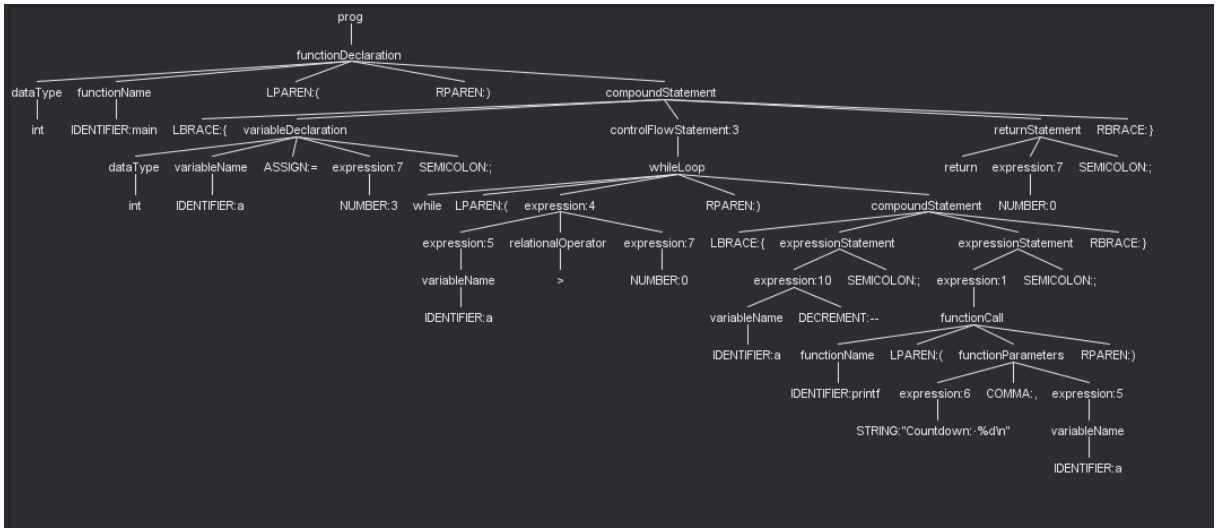

Figure 6.4: Parse tree for While Loop (Zoom the PDF)

6.5 | Function Declaration and Call:

```
int add(int x, int y) {
    return x + y;
}

int main() {
    int result = add(3, 4);
    printf("Result: %d\n", result);
    return 0;
}
```

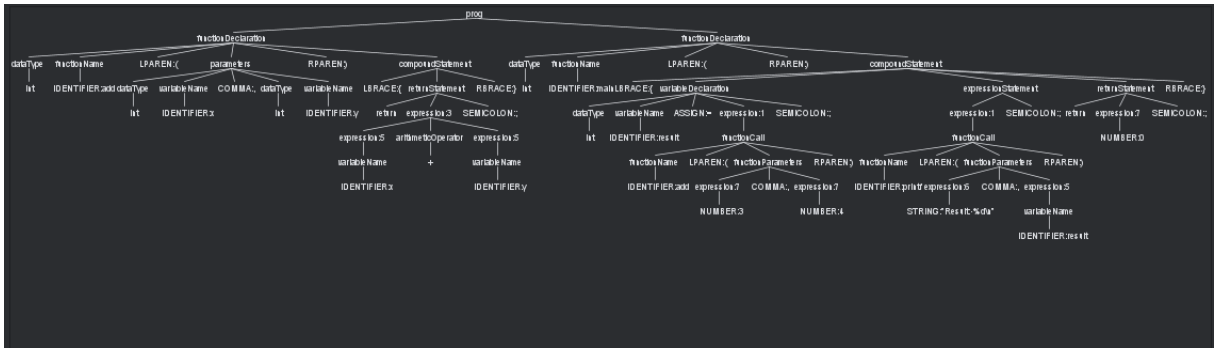

Figure 6.5: Parse tree for Function Declaration and Call (Zoom the PDF)

6.6 | Syntactically Incorrect Input:

```
int main() {
    for (int i = 0; i < 5; i++) {
        printf("%d ", i);
    } // Missing closing brace
```

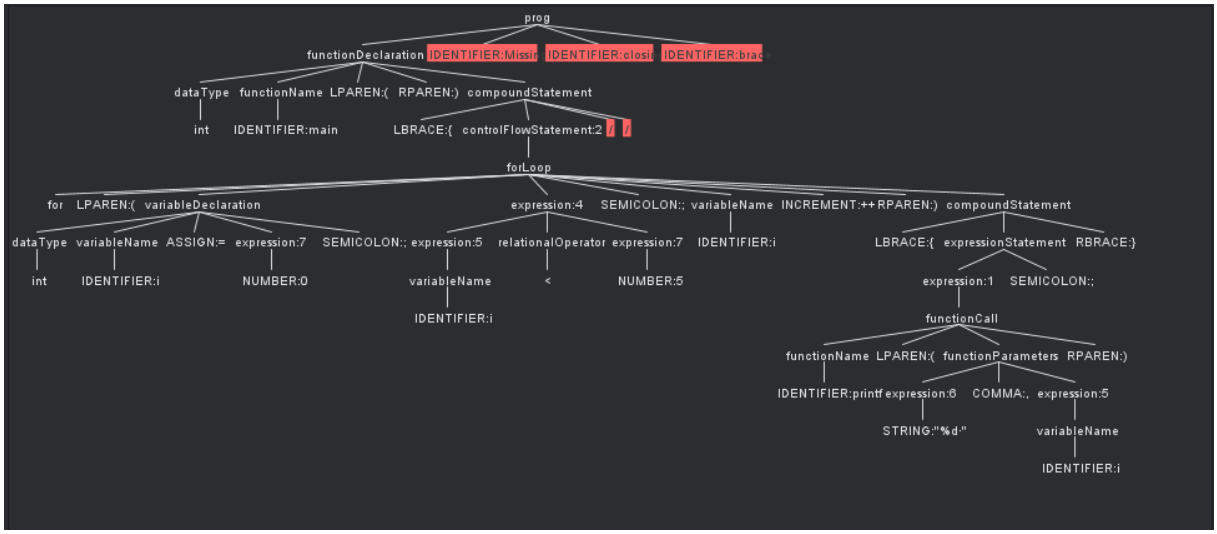

Figure 6.6: Parse tree for Syntactically Incorrect Input (Zoom the PDF)

7 | Conclusion

The grammar[2] is comprehensive and covers various aspects of a programming language, including declarations, control flow, expressions, and functions. It is suitable for generating parsers and compilers for a language with C-like syntax. The grammar also handles whitespace gracefully, improving readability. Further development or adjustments can be made based on specific language requirements.

8 | References

[1] IntelliJ IDEA. *https://www.jetbrains.com/idea/download/*.

[2] Fajla Rabby Khan. *https://github.com/fajlarabbykhan/Context-Free-Grammar-CFG-and-Implementation-Using-ANTL tree/main*.