BLUE LANGUAGE

By

Group Blue

Vi Phan
Joe Huang
Vuong Nguyen
Tri Pham

Professor Ron Mak

Fall 2018
San Jose State University

I. Introduction

In this project, the programming language called Blue that we created is a top-down programming language, and it is similar to C. Like C language, the item types need to be specified before declaring a variable. The Blue language can perform basic arithmetic operations between two variables, and it also has program flow control such as a conditional and looping statement. Similar to most programming languages out there, Blue supports function call and is able to print on the console. The main idea of this project is to build a compiler which generates code for Java Virtual Machine. Students were asked to develop their own grammar file, and feed it to ANTLR compiler-compiler. ANTLR then generates a parser and a scanner based on the grammar file. By overriding the default visitor methods, we were able to decide what functions to perform when each node was visited. After finishing to overwrite the visitor methods, the compiler was built and run to generate Jasmin object code. Jasmin assembler was used to translate Jasmin assembly language programs to .class file that can run on our target machine, Java Virtual Machine. Lastly, the sample programs written in our own language was compiled and run, and the output result was examined.

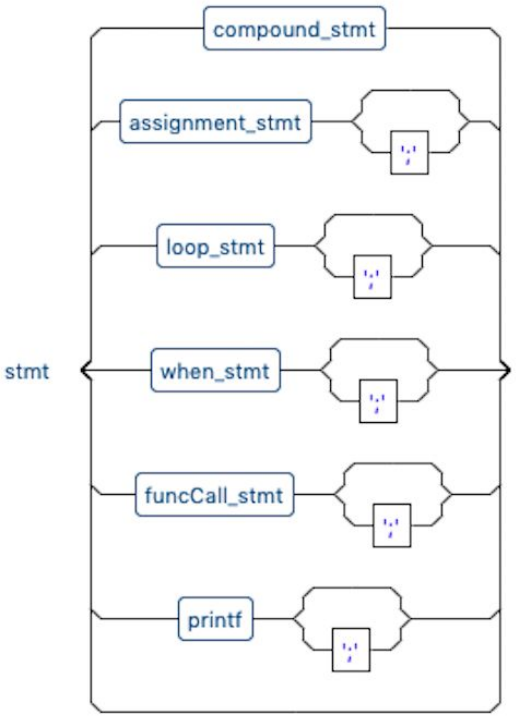
II. Language Construct Implementation

In this project, our team succeeded to complete most of the basic requirement for the compiler as well as building some basic language construct for Blue language. Blue language construct implementation includes:

1. **Declaration**:

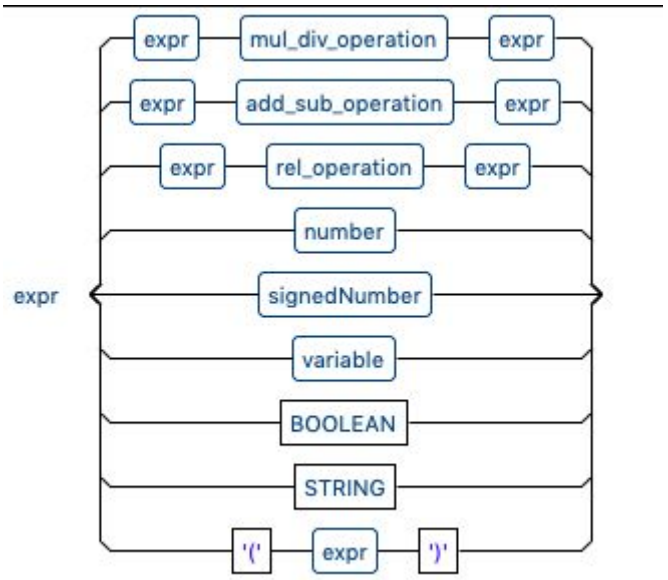
Declaration is implemented for declaring variables. The declaration requires to have the data type at the beginning and following by the variables list which is the list of the variable name. Visitors will enter the declarations node in the tree to create symbol table entry for each variable and push it in the symbol table stack.

2. Statement:



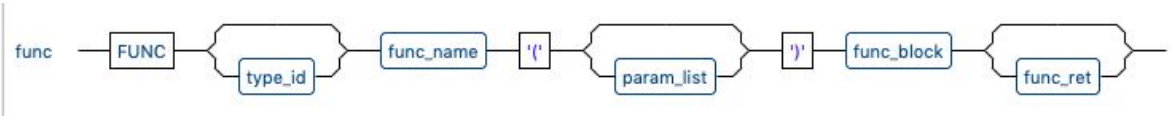
Each statement in the Blue language ends with a semicolon. Our language includes compound statement, assignment statement, loop statement, when statement, function call statement, and printf statement. The WHEN is the conditional statement, and the LOOP is the loop statement in the Blue language. The assignment statement and compound statement are similar to statement for Pascal. However, the compound statement inherit the characteristic with the "{" and "}" as the BEGIN and END. Moreover, beside compound statement, other statement includes the semicolon at the end. It is, unlike C++ and Java, WHEN and LOOP still need a semicolon to signal the end of statements.

3. Expression:



Blue language's expression including expressions for basic operations, relational operations, signed number, numbers, variables, and strings. Even though the boolean is listed as an expression. Due to time constraint, boolean is not implemented. Since the expression node is able to return the type of that expression, a type checking is implemented to ensure it follows that overall rules of the operation where the restriction is that no different type can be operated.

4. Function:



Blue language provides function creation. In order to declare a function, they must include FUNC in the beginning as the keyword to identify a function; then, followed by return type, the name of the function, and a parameter list which is enclosed by parenthesis. Parameters are separated by commas. Blue language only provides pass by value; however, the team is considered to add pass by reference in the future. The func_block is the definition of a function. The func_block consists for local variable declaration and the compound statement for the function.

5. Main:

Similar to C++ and Java, Blue language consists of a main function where it does the operation for the whole source program. The construct of main includes the compound statement and word "MAIN" to separate from the other normal function.

6. Overall Construct:

Blue language uses the inspiration from Pascal where at the end of a program, there is a period to indicate the end of a source code. In Blue language, a “START” is used to signal the beginning of main source code and “TERMINATE” indicate the end.

Basic requirement completed:

- Data types with type checking
- Basic arithmetic operations with operator precedence
- Assignment statements
- One conditional control statement
- One looping control statement
- Functions with calls and returns.
- Parameters passed by value.
- Basic error recovery.

The team was able to complete the following basic requirement:

- Able to compile for 2 data types and check if the data type is matched. If not, a console output will be printed on the Eclipse IDE.
- Able to implement basic +,-,*,/,(), and relational operator.
- Able to assign values for variables.
- Able to run a WHEN conditional statement.
- Able to complete a LOOP loop statement.
- Able to call function and return a value
- Able to pass a value to a function
- Able to detect error and continue to check the file.

III. Result

During this project, we encountered several difficulties. The main challenge was that it was pretty hard to even get the environment setup on all the team members’ PC. Most team members used Windows as their only OS, yet ANTLR was not compatible with Windows OS very well. Therefore, lots of time and efforts were spent on debugging compiling and execution errors. Fortunately, we were able to get through of that and get a good result.

For the sample program (Figure 1), the Jasmine Assembly codes were generated correctly according to the Professor’s code templates (Figure 2 through 7).

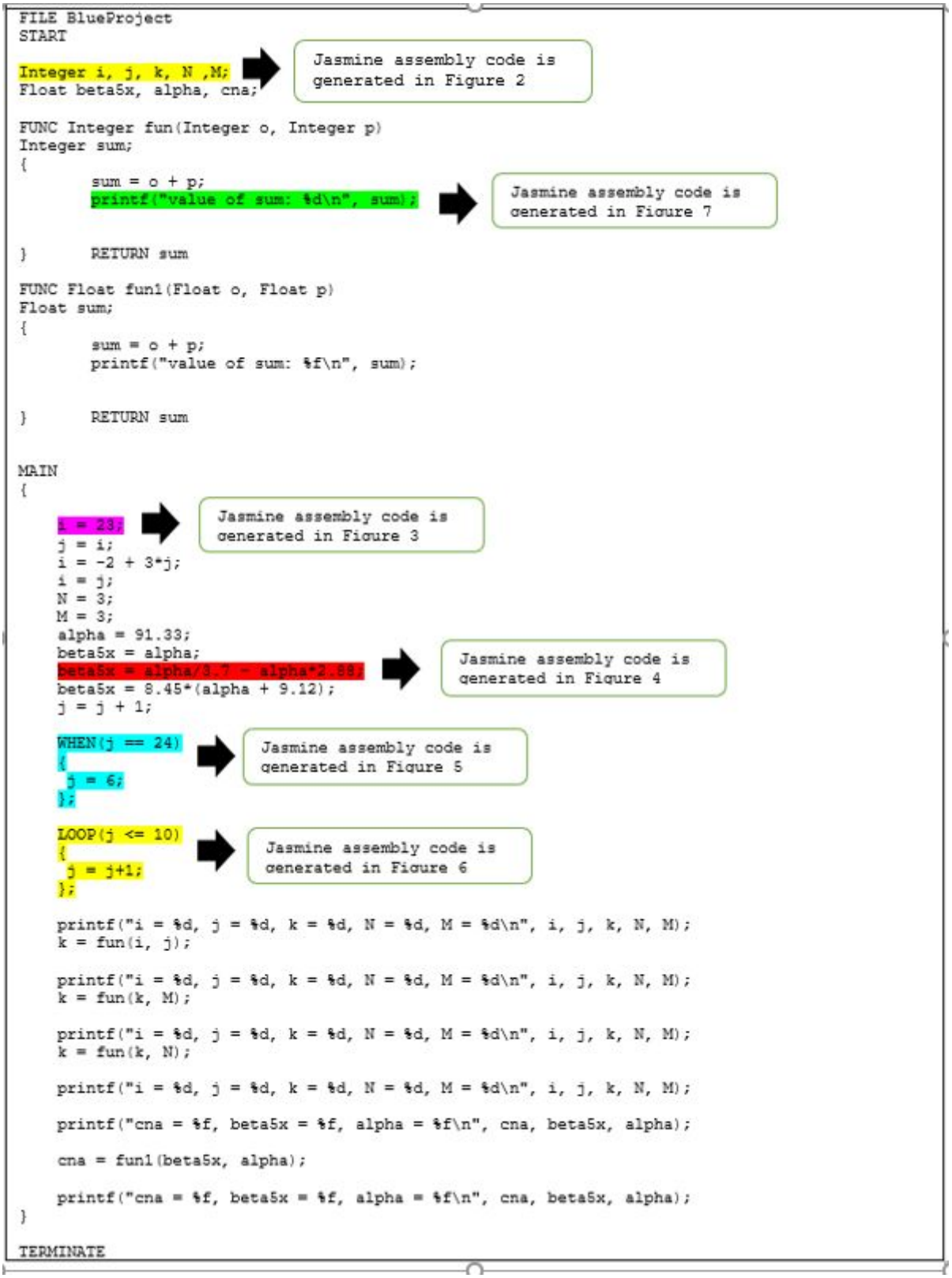


Figure 1. A Sample program that was Written by Blue Language

```
.field private static i I
.field private static j I
.field private static k I
.field private static N I
.field private static M I
.field private static beta5x F
.field private static alpha F
.field private static cna F
```

Figure 2. Jasmine Assembly Code for Declaration Section

```
; i=23

    ldc 23
    putstatic   BlueProject/i I
```

Figure 3. Jasmine Assembly Code for Assignment Statement

```
; beta5x=8.45*(alpha+9.12)

    ldc 8.45
    getstatic   BlueProject/alpha F
    ldc 9.12
    fadd
    fmul
    putstatic   BlueProject/beta5x F
```

Figure 4. Jasmine Assembly Code for Basic arithmetic operations

```
; WHEN(j==24){j=6;}


; WHEN(j==24){j=6;}

    getstatic   BlueProject/j I
    ldc 24
    if_icmpeq   L000
    iconst_0
     goto   L001
L000:
    iconst_1
L001:
    ifeq    L002

; j=6

    ldc 6
    putstatic   BlueProject/j I

L002:
```

Figure 5. Jasmine Assembly Code for WHEN Statement

```
; LOOP(j<=10){j=j+1;}


L003:


; LOOP(j<=10){j=j+1;}

    getstatic   BlueProject/j I
    ldc 10
    if_icmple   L004
    iconst_0
     goto   L005
L004:
    iconst_1
L005:
    ifeq    L006

; j=j+1

    getstatic   BlueProject/j I
    ldc 1
    iadd
    putstatic   BlueProject/j I

    goto    L003
```

Figure 6. Jasmine Assembly Code for the LOOP Statement

```
.limit locals 1
.limit stack 1
.end method


.method private static fun(II)I

    iload_0
    iload_1
    iadd
    istore_2


;printf("value of sum: %d\n",sum)
    getstatic    java/lang/System/out Ljava/io/PrintStream;
    ldc          "value of sum: %d\n"
    iconst_1
    anewarray    java/lang/Object
    dup
    iconst_0
    iload_2
    invokestatic  java/lang/Integer.valueOf(I)Ljava/lang/Integer;
    aastore
    invokevirtual java/io/PrintStream.printf(Ljava/lang/String;[Ljava/lang/Object;)Ljava/io/PrintStream;
    pop


    iload_2
    ireturn
```

Figure 7. Jasmine Assembly Code for the Function and Printf

IV. Instructions

- Generate Parser and Lexer using antlr4 in terminal.
    - java -jar /antlr4-4.7.1/antlr-4.7-complete.jar -no-listener -visitor -Dlanguage="Cpp" filename.g4
- Include the header for boost and antlr4-runtime header file.
- Include library antlr4-runtime.
- Build and run the main file to generate jasmine file.
- Using jasmine assembler in the command line to generate .class file.
    - java -jar jasmin.jar ProjectFileName

  Where the jasmine.jar path needed to be included
  For example:
  "java -jar /Users/viphan/Desktop/jasmin-2.4/jasmin.jar BlueProject.j"
- Using Pascal runtime given by the professor to run the sample code output and runtime output.
    - java -cp .:PascalRTL.jar BlueProject

  The BlueProject is .class file and PascalRTL is jar file given by the professor. PascalRTL should be in the project folder to run the command line.

V. Conclusion

Overall, we successfully compiled and ran the code. All the requirements that we need to do for our project compiler have been tested and verified successfully. Although this project has been a challenging one and also complicated, our group managed to finish the compiler in

time. We did encounter some errors and problems during conducting this project, but we managed to fix all of them one by one. What's great about this project was they gave us a better understanding about designing a new language compiler after overcoming all those challenge tasks. Overall, doing this project has been a great experience for our group, and in the future, we could use that knowledge in designing a new effective compiler to contribute to our engineering industry.