Indian Institute of Information Technology Vadodara
Government Engineering College, Sector-28, Gandhinagar,
Gujarat – 382028

Compiler Design Project
Report - 2

Slate

Submitted by

Jaya Sharma (201551021)
Kriti Singhal (201551024)
Prathmesh Bapat (201551075)
Sudhanshu Bhardwaj (201551076)

About the Language

Our language is inspired from python. We are trying to implement some basic features of python with different conventions, apart from that we have also included some more features of our own. Our language is basically focused on scientific mathematical calculation.

Features

1. Various inbuilt libraries can be imported.
2. Primitive data types : int, float, boolean, rational numbers, char .
3. Composite data type : List
4. I/O statements .
5. Arithmetic operators : +, -, *, /, //, **, %
6. Relational operators : <, <=, >, >=, =, !=, !
7. Bitwise operators : |, &, ^, ~
8. Conditional statements : if, else if, else and switch statements .
9. Two types of Iterative statements.
10. BODMAS rule is followed in arithmetic operations.
11. Control statements : break, continue, breakAll .

Notations

- " : " is used for assignment and " = " for equality check.
- " / " is used for division whereas " // " is used for floor division.

- “ ** ” is used as power operator.
- “ ^ ” is used as bitwise xor operator.
- Rest of the operators are same as in python.

Tokens used in grammar :

IMPORT COMMENTS STRING LETTER DIGIT CHARACTER LOOP BREAK CONTINUE BREAK_ALL IF ELSE ELSE_IF RETURN SWITCH CASE TRUE FALSE DEFAULT

Grammar -->

```
program  :   import_stmts statement
           | import_stmts func_def
           | function_definition func_def
           | function_definition statement
           | statement
           | statement func_def
           | ϵ


import_stmts  :  import_stmts import_stmt
               | import_stmt

import_stmt    :   IMPORT inbuilt_function

inbuilt_function  :  inbuilt_function, identify
                   | identify

statements  :  function_call
             | assignment_stmts
             | iterative_stmts
             | control_stmts
             | conditional_stmts
             | return_stmts
             | COMMENTS
             | switch_stmts
```

```
statement   :   statements statement
                | statements

function_call  :  identify -> actual


actual  :  actual, parameter
           | parameter

parameter  :  STRING
              | num
              | identifier
              | ϵ

identify   :   LETTER optionNew

optionNew     :   DIGIT optionNew
                  |  LETTER optionNew
                  |  _ optionNew | LETTER
                  |  DIGIT
                  |  _
                  | ϵ

identifier    :    identify
                  | identify[index]

index   :   num
            | identifier
            | for_arithmetic

function_definition    :    identify (formal) { statements }

func_def   :   function_definition func_def
               | function_definition

formal   :   formal, identify
             | identify
             | ϵ
```

```
assignment_stmts  :  identifier : value
value   :   arithmetic_exp
            | boolean_exp
            | num
            | identifier
            | CHARACTER
            | list
            | STRING

iterative_stmts   :   LOOP (for_assignment ; boolean_exp ; for_assignment) { statements }
                      | LOOP (boolean_exp) { statements }

for_assignment   :   identifier :  for_value

for_value    :    num
                  | identifier
                  | CHARACTER
                  | for_arithmetic

for_arithmetic   :   for_A
                     | ( for_arithmetic )

for_A    :   for_A + for_B
             | for_A – for_B
             | for_B
             | (for_A)

for_B   :   for_B * for_C
            | for_B / for_C
            | for_B % for_C
            | for_C
            | ( for_B )

for_C   :   for_D ** for_C
            | for_D
            | (for_C)

for_D   :   num
            | identifier
```

```
control_stmts    :    BREAK
                      | CONTINUE
                      | BREAK_ALL

conditional_stmts   :   IF boolean_exp { statements } optional

optional   :   ELSE_IF boolean_exp  { statements } optional  optional2
               | optional2
               | ϵ

optional2    :    ELSE { statements }
                  | ϵ

boolean_exp   :   boolean_exp  | |  literal
                  | literal

literal   :   literal  &&  word
              | word

word   :   number < Z
           | number <= Z
           | number > Z
           | number >= Z
           | number =Z
           | number != Z
           | TRUE
           | FALSE
           | ! word

number   :   identifier
             | num
             | arithmetic_exp
             | CHARACTER

Z   :   identifier
        | num
        | arithmetic_exp
```

```
            | CHARACTER

arithmetic_exp   :   arithmetic_exp | term
                     | term
                     | ( arithmetic_exp )

term    :   term ^ X
            | X
            | ( term )

X    :    X & Y
          | Y
          | (X)

Y    :   Y + A
         | Y - A
         | A
         | (Y)

A   :   A * B
        | A / B
        | A % B
        | B
        | (A)

B    :    C ** B
          | C
          | (B)

C    :   ~ num
         | (C)
         | num
         | identifier

return_stmts    :   RETURN result
                    | RETURN result , result

result   :   identifier
             | num
```

```
                | CHARACTER
                | STRING

switch_stmts   :   SWITCH identifier { case_stmts }

case_stmts   :   CASE identifier { statements } case_stmts
                 | DEFAULT  identifier { statements }

list   :   [ phrase ]

phrase   :   name, phrase
                | name
                | ϵ

name   :   num
             | CHARACTER
             |  identifier

num   :   digit1
            | digit1.digit2

digit1   :   DIGIT digit1
              | DIGIT

digit2   :   DIGIT digit2
              |  DIGIT
```

****

Note : The code is not running correctly, there is some error in parsing. We are working on it and hope to present the correct, modified code while demonstration of the project.