Shiraz University

Python to C Project Document

By Alireza Davoodi & Abdolrahim Touranian

Under the guidance of Professor Eghbal Mansoori

Lexical Analyzer

The lexical analyzer uses a lex file that defines simple instructions: assignments, if-else, while, and for blocks. Arithmetic and comparative expressions including variables and constants are supported. A short example of our lexical analyzer is as follow:

- `"#"(.)*` to detect inline comments
- `[\t" "]*` to ignore whitespaces
- `-?(([0-9]+)|([0-9]*\.[0-9]+)([eE][-+]?[0-9]+)?)` to detect real numbers
- `[_a-zA-Z][_a-zA-Z0-9]*` to detect identifiers
- "if" {debug("If"); return T_If;}
- "in" {debug("In"); return T_In;}
- "range" {debug("Range"); return T_Range;}
- "for" {debug("For"); return T_For;}
- "while" {debug("While"); return T_While;}
- "and" {debug("And"); return T_And;}
- "or" {debug("Or"); return T_Or;}
- "not" {debug("Not"); return T_Not;}
- "elif" {debug("Elif"); return T_Elif;}
- "else" {debug("Else"); return T_Else;}
- "{" {debug("OB"); scope_depth++; return T_Cln;}
- "}" {debug("CB"); scope_depth--; return T_CB;}
- ">" {debug("GT"); return T_GT;}
- "<" {debug("LT"); return T_LT;}
- ">=" {debug("EGT"); return T_EGT;}
- "<=" {debug("ELT"); return T_ELT;}
- "==" {debug("EQ"); return T_EQ;}
- "!=" {debug("NEQ"); return T_NEQ;}
- "True" {debug("True"); return T_True;}
- "False" {debug("False"); return T_False;}
- "+" {debug("PL"); return T_PL;}
- "-" {debug("MN"); return T_MN;}
- "*" {debug("ML"); return T_ML;}

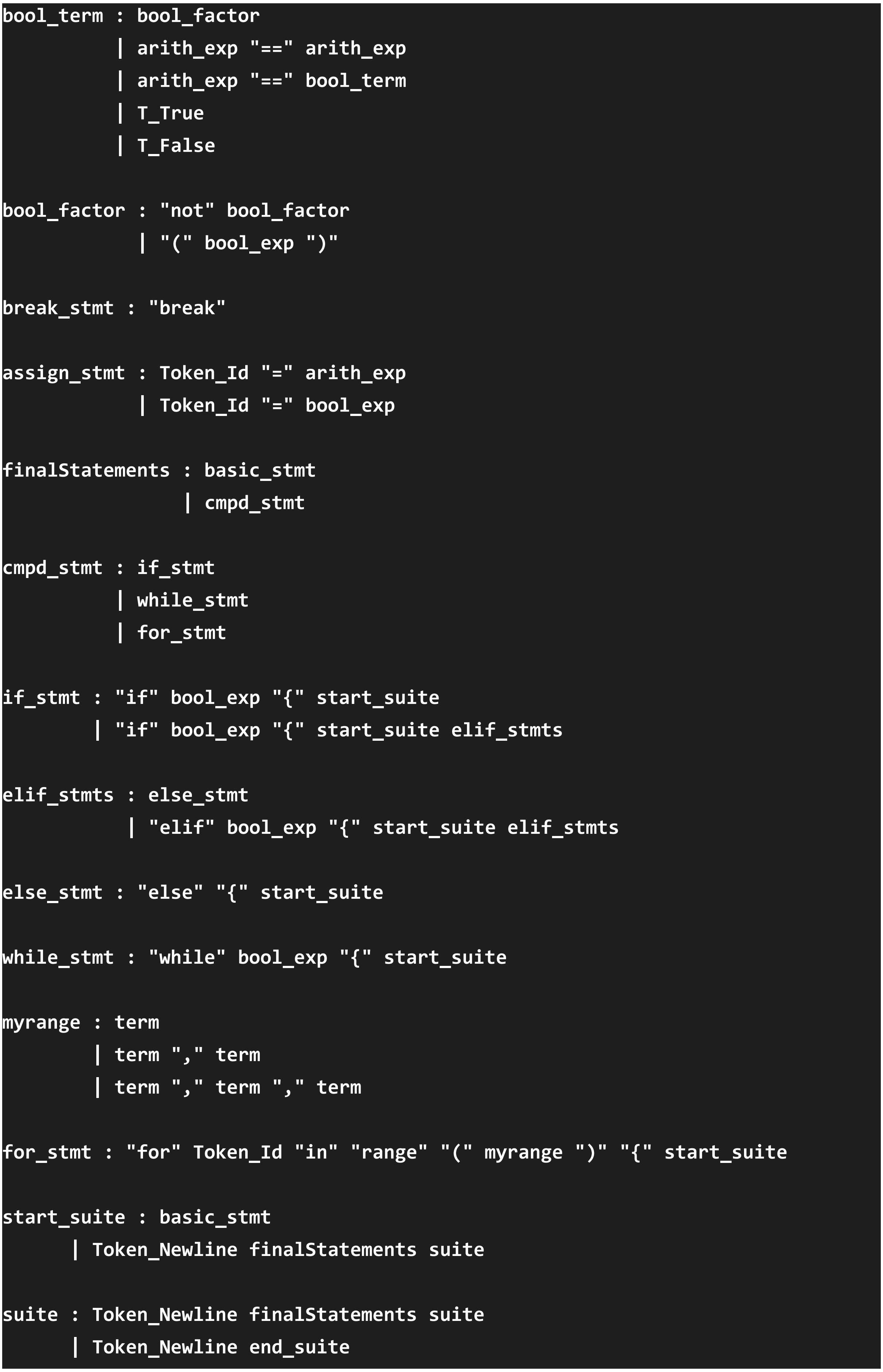
- "/" {debug("DV"); return T_DV;}
- "(" { debug("OP"); return T_OP;}
- ")" {debug("CP"); return T_CP;}
- "[" {debug("OB"); return T_OB;}
- "]" {debug("CB"); return T_CB;}
- "," {debug("Comma"); return T_Comma;}
- "=" {debug("EQL"); return T_EQL;}

Keywords if, while, … are also defined as tokens in the lexical analyzer. All the definitions could be viewed in the lex.l file.

Yacc
We have implemented the below grammar in the yacc file (translator.y) to generate a syntax tree and three-address code. We then eliminate unused variables. Finally, we convert the final three-address to C code (output.c)

Grammar

```
constant : Number

term : Token_Id | constant

StartParse : Token_Newline StartParse | finalStatements Token_Newline
StartParse | finalStatements Token_Newline

basic_stmt : break_stmt
           | assign_stmt
           | arith_exp
           | bool_exp

arith_exp : term
          | arith_exp  "+"  arith_exp
          | arith_exp  "-"  arith_exp
          | arith_exp  "*"  arith_exp
          | arith_exp  "/"  arith_exp
          | "-" arith_exp
          | "(" arith_exp ")"

bool_exp : bool_term "or" bool_term
         | arith_exp "<" arith_exp
         | bool_term "and" bool_term
         | arith_exp ">" arith_exp
         | arith_exp "<=" arith_exp
         | arith_exp "<=" arith_exp {$$ = createOp(">=", 2, $1, $3);}
         | bool_term {$$=$1;};
```



```
bool_term : bool_factor
          | arith_exp "==" arith_exp
          | arith_exp "==" bool_term
          | T_True
          | T_False

bool_factor : "not" bool_factor
            | "(" bool_exp ")"

break_stmt : "break"

assign_stmt : Token_Id "=" arith_exp
            | Token_Id "=" bool_exp

finalStatements : basic_stmt
                | cmpd_stmt

cmpd_stmt : if_stmt
          | while_stmt
          | for_stmt

if_stmt : "if" bool_exp "{" start_suite
        | "if" bool_exp "{" start_suite elif_stmts

elif_stmts : else_stmt
           | "elif" bool_exp "{" start_suite elif_stmts

else_stmt : "else" "{" start_suite

while_stmt : "while" bool_exp "{" start_suite

myrange : term
        | term "," term
        | term "," term "," term

for_stmt : "for" Token_Id "in" "range" "(" myrange ")" "{" start_suite

start_suite : basic_stmt
      | Token_Newline finalStatements suite

suite : Token_Newline finalStatements suite
      | Token_Newline end_suite
end_suite : "}" finalStatements
          | "}"
          | epsilon
```

Symbol Table

Each item in the symbol table consists of the following properties:

- Name/Value
- Scope
- Type
- Declaration Line

Constants, Identifiers, labels, and temporaries are stored in the symbol table

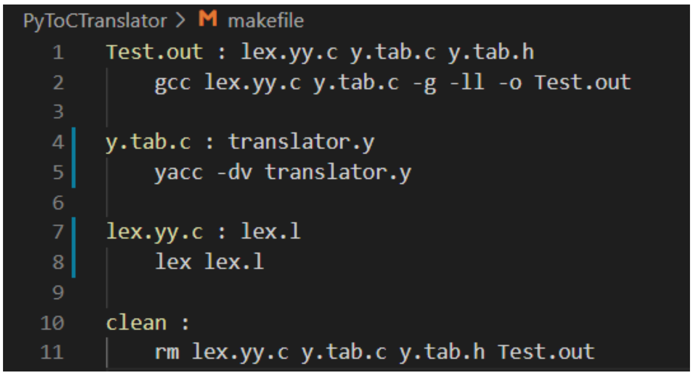
Three-Address code

The code is generated by traversing the syntax tree. We also eliminate unused variables in any block after generating the code to do this we eliminate any variable that wasn't used in a not already removed right value.

Error handling

We throw an error specifying line and column if there is a syntax error. We also throw an Undeclared variable error if an undeclared variable is referenced in the scope.

User Manual



```
Test.out : lex.yy.c y.tab.c y.tab.h
    gcc lex.yy.c y.tab.c -g -ll -o Test.out

y.tab.c : translator.y
    yacc -dv translator.y

lex.yy.c : lex.l
    lex lex.l

clean :
    rm lex.yy.c y.tab.c y.tab.h Test.out
```

- **Ubuntu**
  We included a makefile which compiles the c files and generates the Test.out file The Test.out binary file could be executed using GCC, use your ubuntu terminal/bash in order to run the following commands If you encounter any errors, Try to restart from the first command of section “Compilation of Binary and Executable file”.

  1. Preinstallation:

- ○ $ sudo apt install flex bison

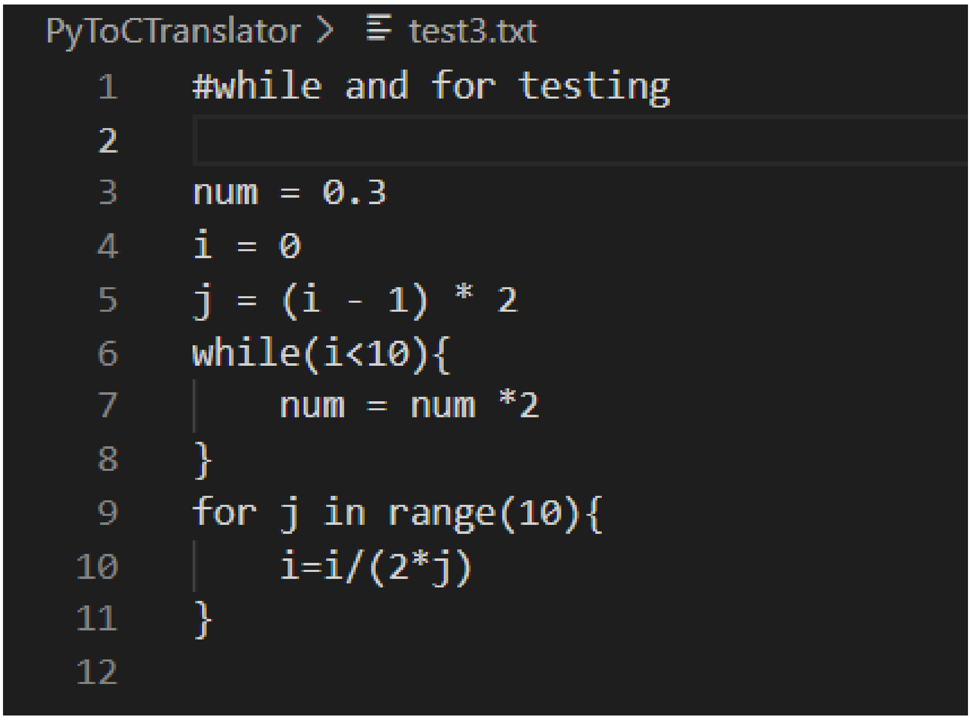
2. Compilation of Binary and Executable file:
   - to delete the old Test.out : $ make clean
   - generate Test.out $ make
   - execute Test.out with input python file test5.txt: $ ./Test.out < test5.txt
   - finally you can test C file with the command $ gcc output.c

- **Windows**
  You can simply install [Ubuntu WSL]() from the Microsoft store and run the above commands or you can download and install flex and bison and run them on cmd. However, the project wasn't tested in this environment.

Examples

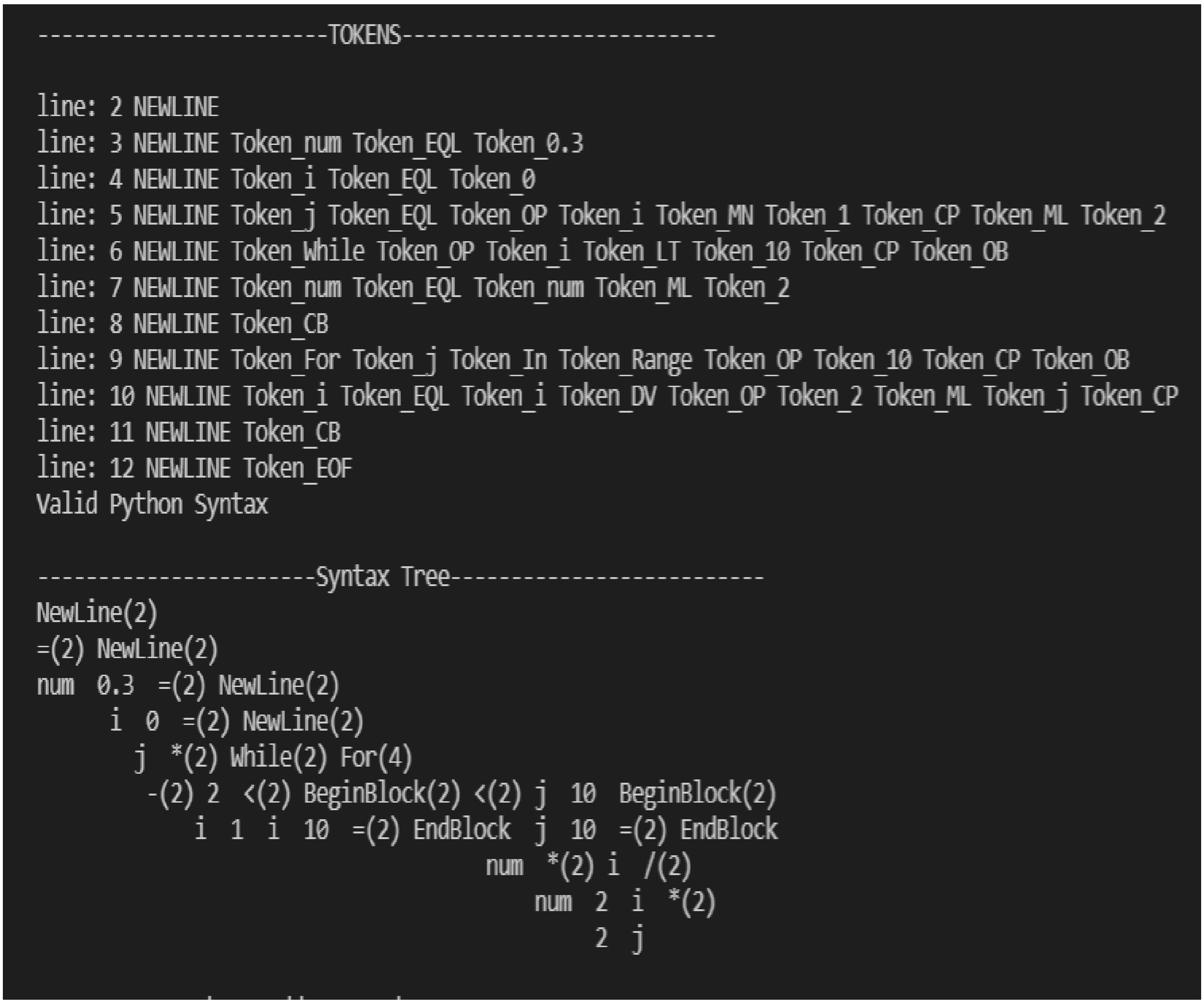
We have implemented 5 test cases here are two important ones, it is good to say you can test other cases with the above command e.g. $ ./Test.out < test3.txt.

Input

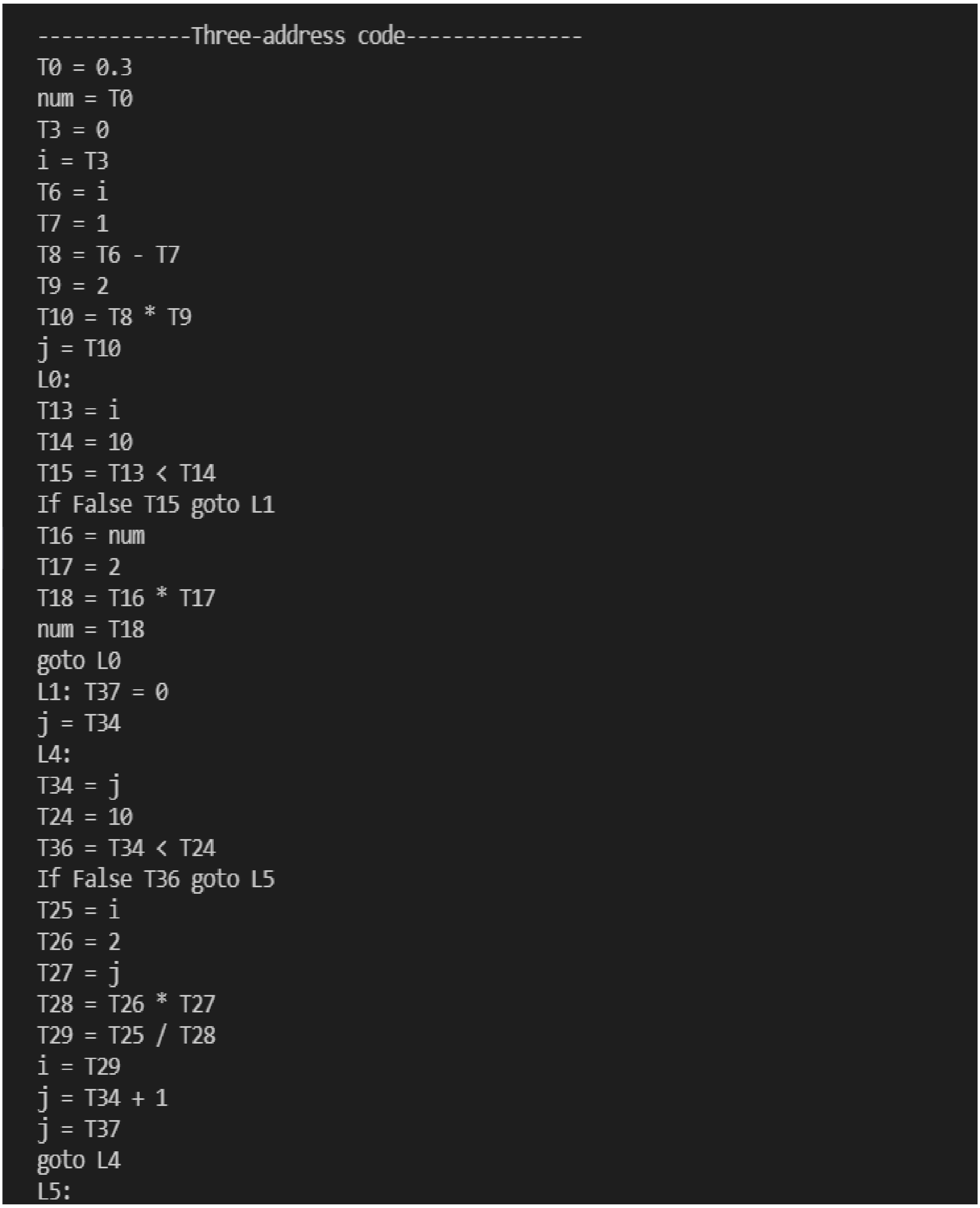


```
#while and for testing

num = 0.3
i = 0
j = (i - 1) * 2
while(i<10){
    num = num *2
}
for j in range(10){
    i=i/(2*j)
}

```

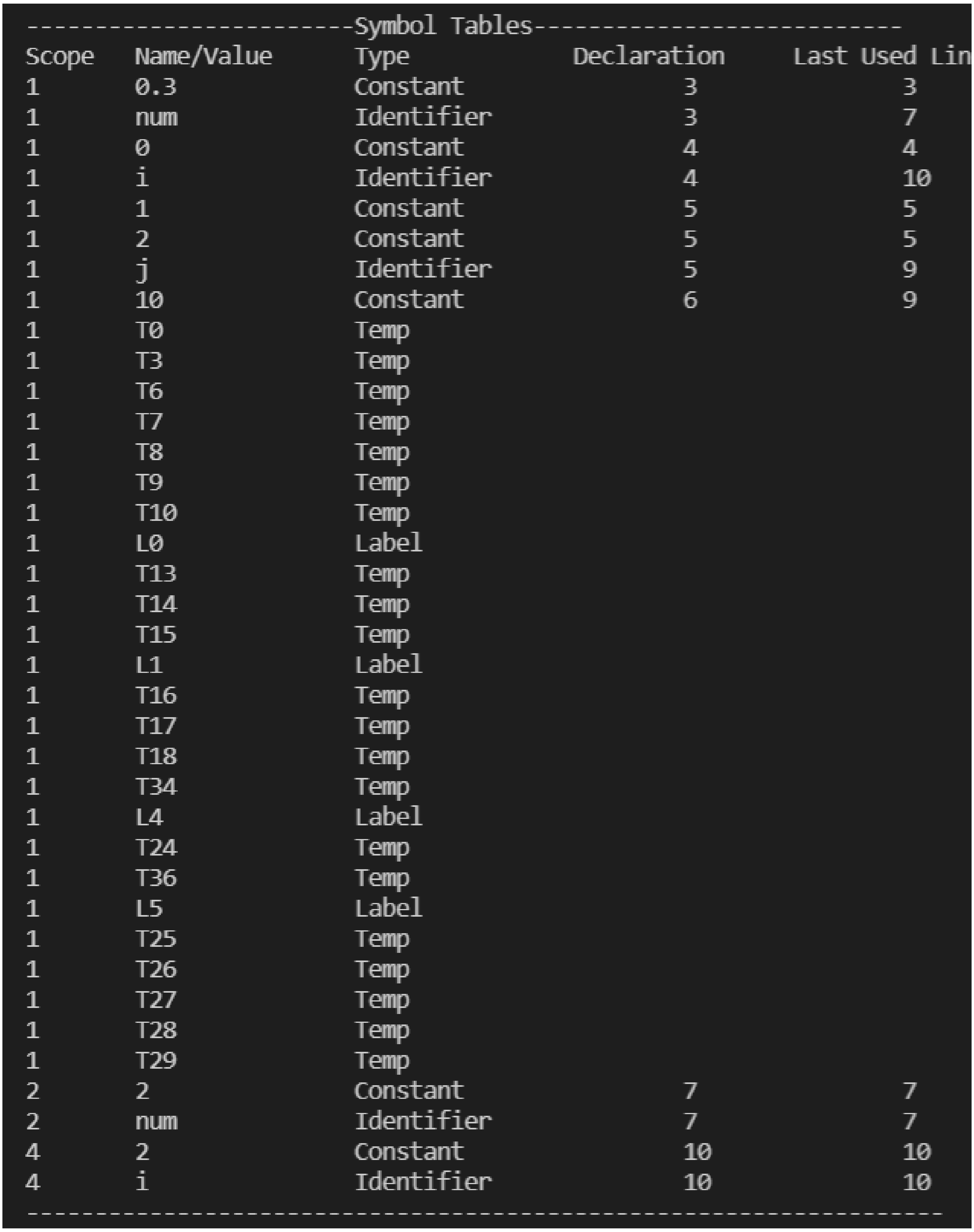
Output



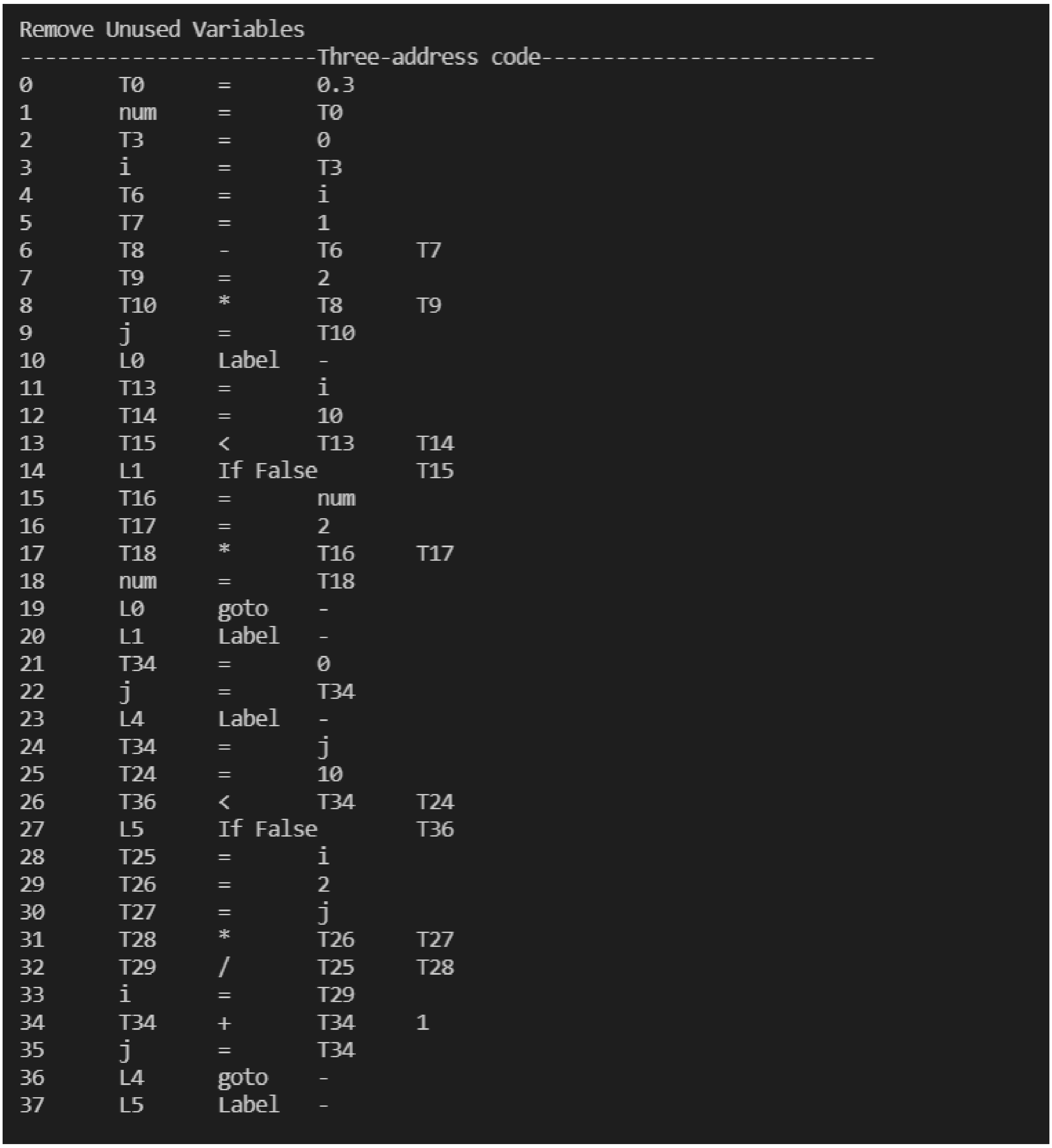
```
------------------------TOKENS--------------------------

line: 2 NEWLINE
line: 3 NEWLINE Token_num Token_EQL Token_0.3
line: 4 NEWLINE Token_i Token_EQL Token_0
line: 5 NEWLINE Token_j Token_EQL Token_OP Token_i Token_MN Token_1 Token_CP Token_ML Token_2
line: 6 NEWLINE Token_While Token_OP Token_i Token_LT Token_10 Token_CP Token_OB
line: 7 NEWLINE Token_num Token_EQL Token_num Token_ML Token_2
line: 8 NEWLINE Token_CB
line: 9 NEWLINE Token_For Token_j Token_In Token_Range Token_OP Token_10 Token_CP Token_OB
line: 10 NEWLINE Token_i Token_EQL Token_i Token_DV Token_OP Token_2 Token_ML Token_j Token_CP
line: 11 NEWLINE Token_CB
line: 12 NEWLINE Token_EOF
Valid Python Syntax

-----------------------Syntax Tree--------------------------
NewLine(2)
=(2) NewLine(2)
num  0.3  =(2) NewLine(2)
      i  0  =(2) NewLine(2)
        j  *(2) While(2) For(4)
         -(2) 2  <(2) BeginBlock(2) <(2) j  10  BeginBlock(2)
             i  1  i  10  =(2) EndBlock  j  10  =(2) EndBlock
                                     num  *(2) i  /(2)
                                         num  2  i  *(2)
                                              2  j
```



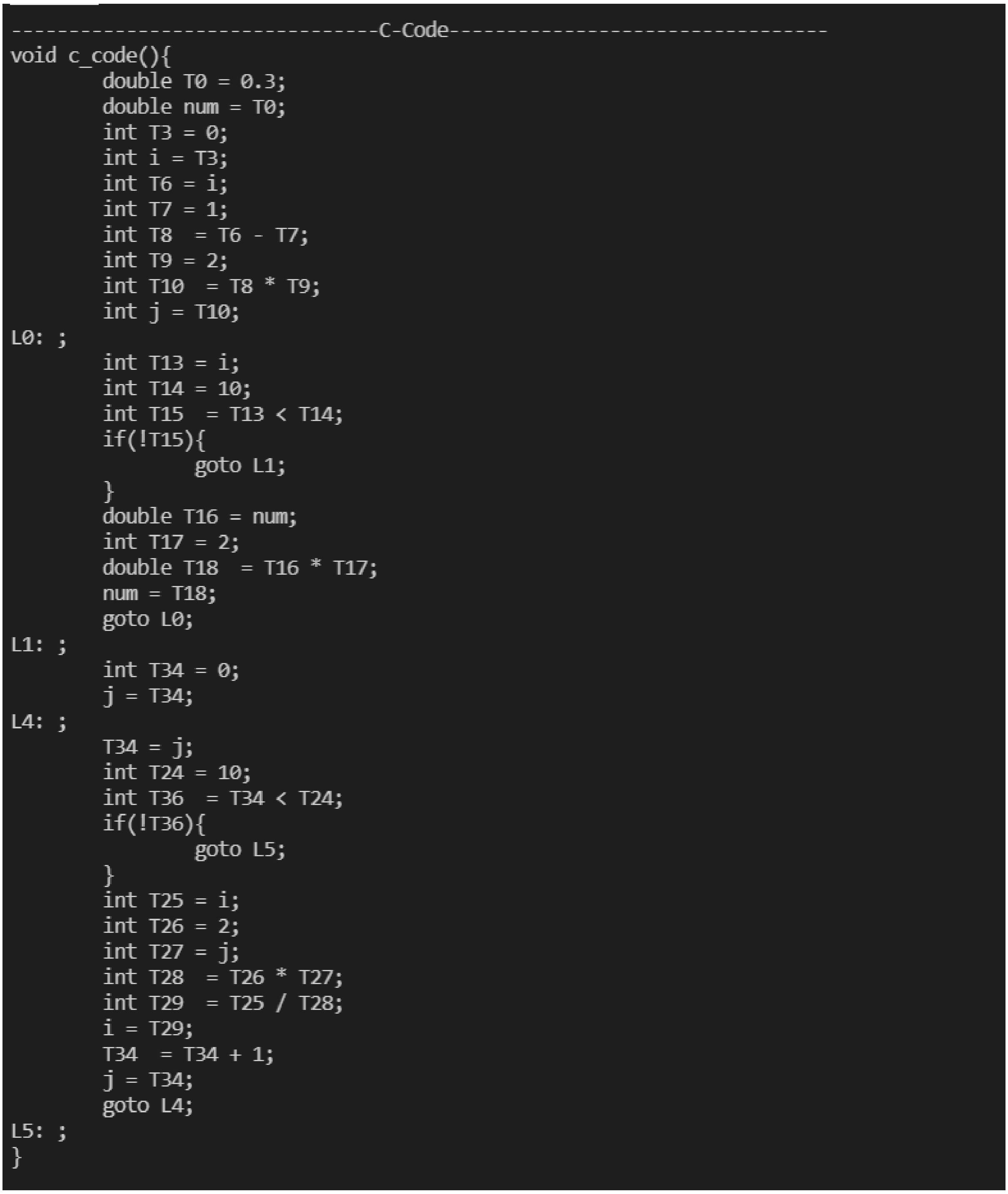
```
-------------Three-address code---------------
T0 = 0.3
num = T0
T3 = 0
i = T3
T6 = i
T7 = 1
T8 = T6 - T7
T9 = 2
T10 = T8 * T9
j = T10
L0:
T13 = i
T14 = 10
T15 = T13 < T14
If False T15 goto L1
T16 = num
T17 = 2
T18 = T16 * T17
num = T18
goto L0
L1: T37 = 0
j = T34
L4:
T34 = j
T24 = 10
T36 = T34 < T24
If False T36 goto L5
T25 = i
T26 = 2
T27 = j
T28 = T26 * T27
T29 = T25 / T28
i = T29
j = T34 + 1
j = T37
goto L4
L5:
```



```
------------------------Symbol Tables---------------------------
Scope   Name/Value      Type            Declaration     Last Used Lin
1       0.3             Constant                3               3
1       num             Identifier              3               7
1       0               Constant                4               4
1       i               Identifier              4               10
1       1               Constant                5               5
1       2               Constant                5               5
1       j               Identifier              5               9
1       10              Constant                6               9
1       T0              Temp
1       T3              Temp
1       T6              Temp
1       T7              Temp
1       T8              Temp
1       T9              Temp
1       T10             Temp
1       L0              Label
1       T13             Temp
1       T14             Temp
1       T15             Temp
1       L1              Label
1       T16             Temp
1       T17             Temp
1       T18             Temp
1       T34             Temp
1       L4              Label
1       T24             Temp
1       T36             Temp
1       L5              Label
1       T25             Temp
1       T26             Temp
1       T27             Temp
1       T28             Temp
1       T29             Temp
2       2               Constant                7               7
2       num             Identifier              7               7
4       2               Constant                10              10
4       i               Identifier              10              10
-------------------------------------------------------------------
```



```
Remove Unused Variables
------------------------Three-address code---------------------------
0       T0      =       0.3
1       num     =       T0
2       T3      =       0
3       i       =       T3
4       T6      =       i
5       T7      =       1
6       T8      -       T6      T7
7       T9      =       2
8       T10     *       T8      T9
9       j       =       T10
10      L0      Label   -
11      T13     =       i
12      T14     =       10
13      T15     <       T13     T14
14      L1      If False        T15
15      T16     =       num
16      T17     =       2
17      T18     *       T16     T17
18      num     =       T18
19      L0      goto    -
20      L1      Label   -
21      T34     =       0
22      j       =       T34
23      L4      Label   -
24      T34     =       j
25      T24     =       10
26      T36     <       T34     T24
27      L5      If False        T36
28      T25     =       i
29      T26     =       2
30      T27     =       j
31      T28     *       T26     T27
32      T29     /       T25     T28
33      i       =       T29
34      T34     +       T34     1
35      j       =       T34
36      L4      goto    -
37      L5      Label   -
```

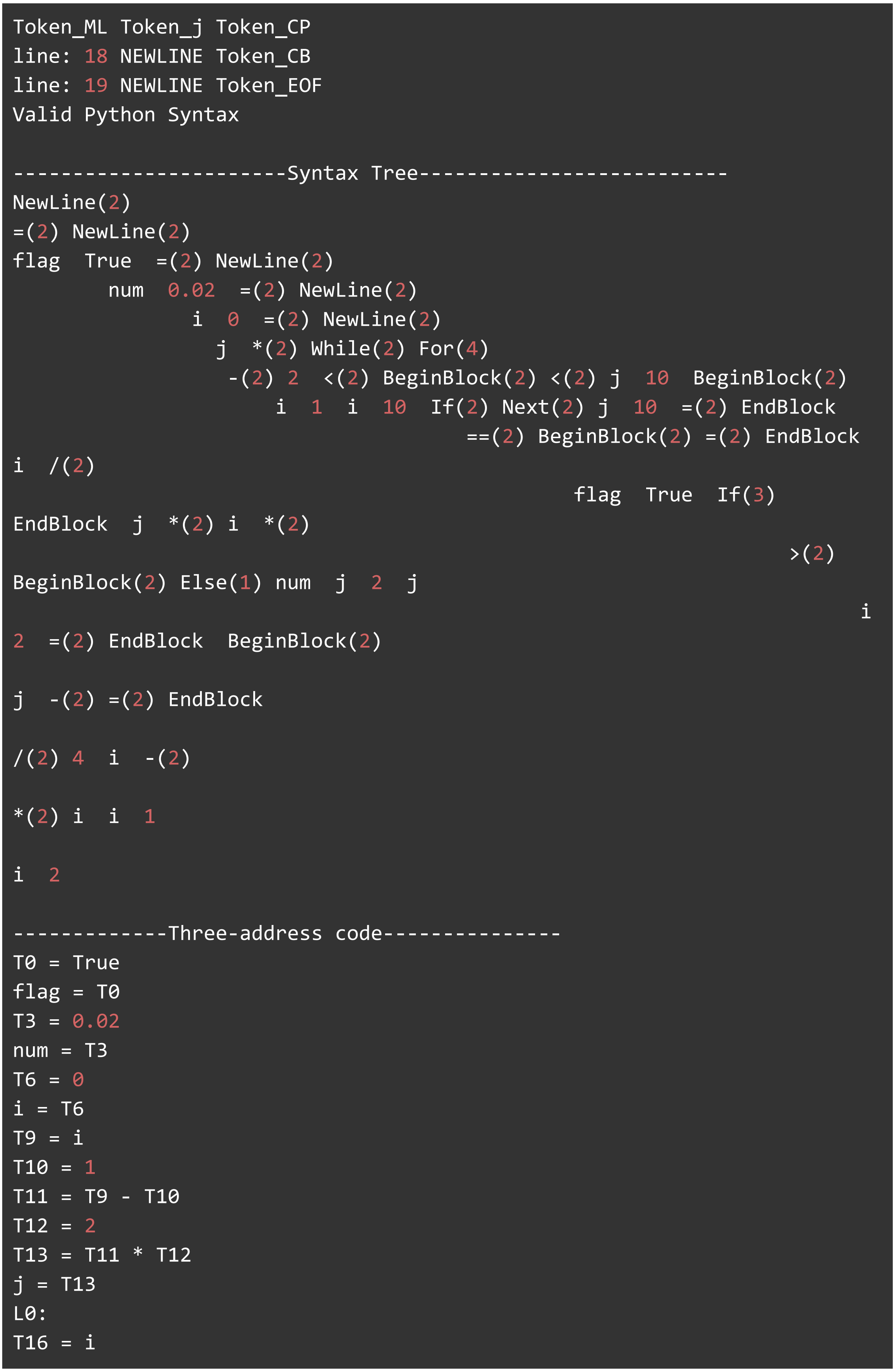


```
-------------------------------C-Code--------------------------------
void c_code(){
        double T0 = 0.3;
        double num = T0;
        int T3 = 0;
        int i = T3;
        int T6 = i;
        int T7 = 1;
        int T8  = T6 - T7;
        int T9 = 2;
        int T10  = T8 * T9;
        int j = T10;
L0: ;
        int T13 = i;
        int T14 = 10;
        int T15  = T13 < T14;
        if(!T15){
                goto L1;
        }
        double T16 = num;
        int T17 = 2;
        double T18  = T16 * T17;
        num = T18;
        goto L0;
L1: ;
        int T34 = 0;
        j = T34;
L4: ;
        T34 = j;
        int T24 = 10;
        int T36  = T34 < T24;
        if(!T36){
                goto L5;
        }
        int T25 = i;
        int T26 = 2;
        int T27 = j;
        int T28  = T26 * T27;
        int T29  = T25 / T28;
        i = T29;
        T34  = T34 + 1;
        j = T34;
        goto L4;
L5: ;
}
```

Example 2

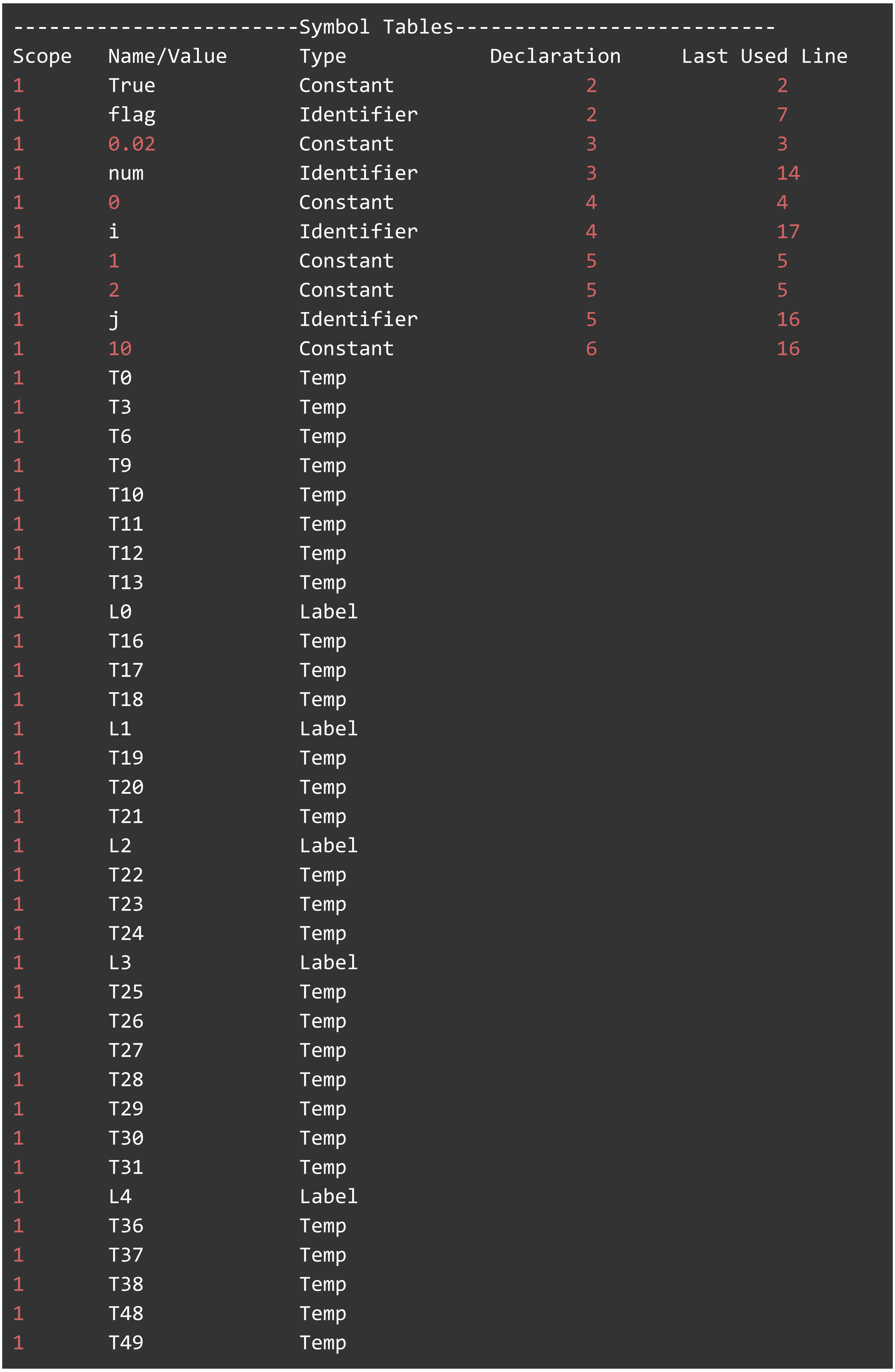
```
#combination of all tests
flag=True
num = 0.02
i=0
j = (i - 1) * 2
while(i<10){
    if( flag == True ){
        if(i > 2){
            j = (i*2)/ i - 4
        }else{
            i = i - 1
        }
    }
    j = num*j
}
for j in range(10){
    i=i/(2*j)
}
```

```
------------------------TOKENS--------------------------

line: 2 NEWLINE Token_flag Token_EQL Token_True
line: 3 NEWLINE Token_num Token_EQL Token_0.02
line: 4 NEWLINE Token_i Token_EQL Token_0
line: 5 NEWLINE Token_j Token_EQL Token_OP Token_i Token_MN Token_1
Token_CP Token_ML Token_2
line: 6 NEWLINE Token_While Token_OP Token_i Token_LT Token_10 Token_CP
Token_OB
line: 7 NEWLINE Token_If Token_OP Token_flag Token_EQ Token_True
Token_CP Token_OB
line: 8 NEWLINE Token_If Token_OP Token_i Token_GT Token_2 Token_CP
Token_OB
line: 9 NEWLINE Token_j Token_EQL Token_OP Token_i Token_ML Token_2
Token_CP Token_DV Token_i Token_MN Token_4
line: 10 NEWLINE Token_CB Token_Else Token_OB
line: 11 NEWLINE Token_i Token_EQL Token_i Token_MN Token_1
line: 12 NEWLINE Token_CB
line: 13 NEWLINE Token_CB
line: 14 NEWLINE Token_j Token_EQL Token_num Token_ML Token_j
line: 15 NEWLINE Token_CB
line: 16 NEWLINE Token_For Token_j Token_In Token_Range Token_OP
Token_10 Token_CP Token_OB
line: 17 NEWLINE Token_i Token_EQL Token_i Token_DV Token_OP Token_2
```

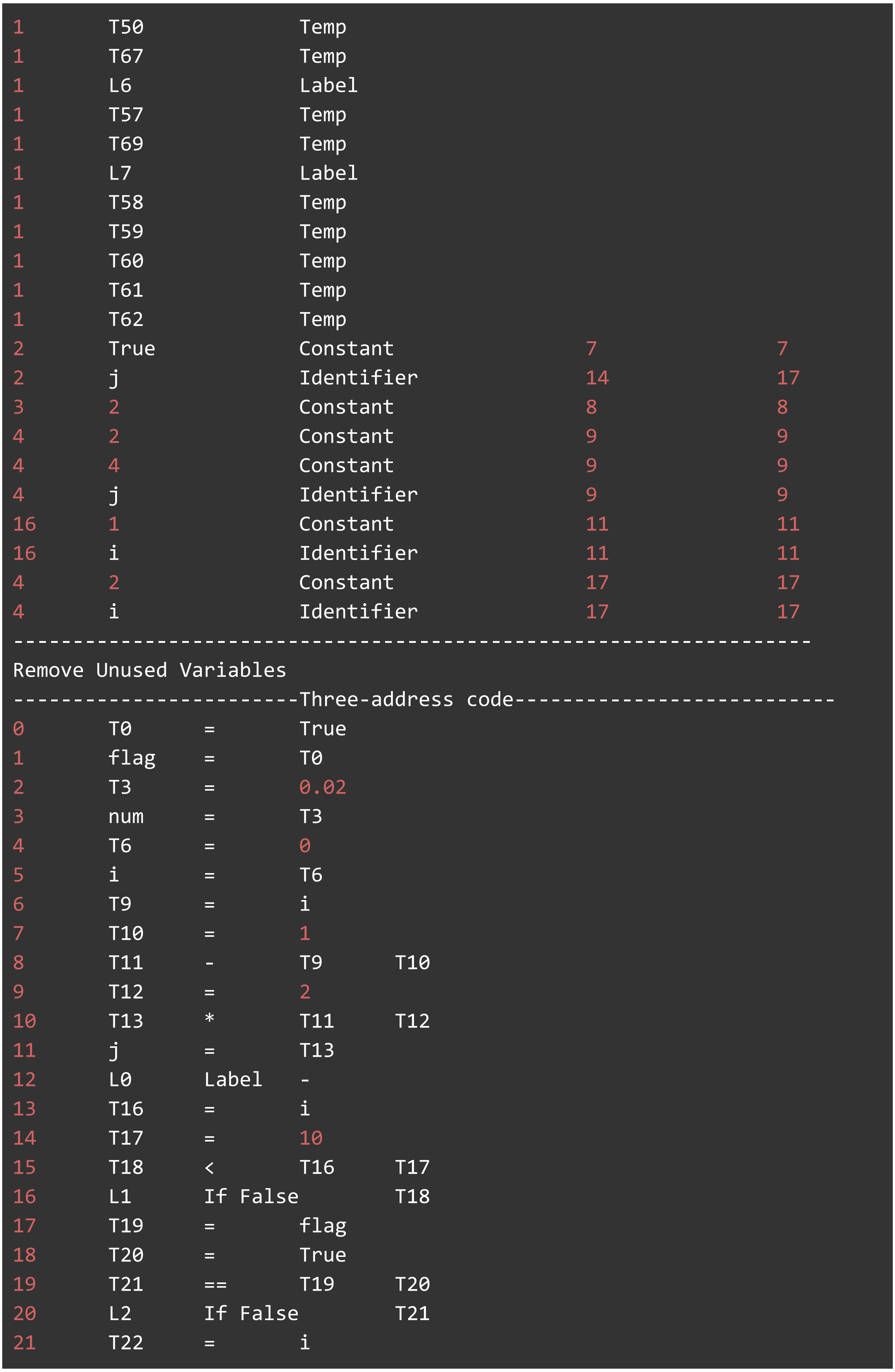


```
Token_ML Token_j Token_CP
line: 18 NEWLINE Token_CB
line: 19 NEWLINE Token_EOF
Valid Python Syntax

-----------------------Syntax Tree--------------------------
NewLine(2)
=(2) NewLine(2)
flag  True  =(2) NewLine(2)
        num  0.02  =(2) NewLine(2)
               i  0  =(2) NewLine(2)
                 j  *(2) While(2) For(4)
                  -(2) 2  <(2) BeginBlock(2) <(2) j  10  BeginBlock(2)
                      i  1  i  10  If(2) Next(2) j  10  =(2) EndBlock
                                      ==(2) BeginBlock(2) =(2) EndBlock
i  /(2)
                                               flag  True  If(3)
EndBlock  j  *(2) i  *(2)
                                                                 >(2)
BeginBlock(2) Else(1) num  j  2  j
                                                                       i
2  =(2) EndBlock  BeginBlock(2)

j  -(2) =(2) EndBlock

/(2) 4  i  -(2)

*(2) i  i  1

i  2

-------------Three-address code---------------
T0 = True
flag = T0
T3 = 0.02
num = T3
T6 = 0
i = T6
T9 = i
T10 = 1
T11 = T9 - T10
T12 = 2
T13 = T11 * T12
j = T13
L0:
T16 = i
```

```
T17 = 10
T18 = T16 < T17
If False T18 goto L1
T19 = flag
T20 = True
T21 = T19 == T20
If False T21 goto L2
T22 = i
T23 = 2
T24 = T22 > T23
If False T24 goto L3
T25 = i
T26 = 2
T27 = T25 * T26
T28 = i
T29 = T27 / T28
T30 = 4
T31 = T29 - T30
j = T31
goto L4
L3: T36 = i
T37 = 1
T38 = T36 - T37
i = T38
L4: L2: T48 = num
T49 = j
T50 = T48 * T49
j = T50
goto L0
L1: T70 = 0
j = T67
L6:
T67 = j
T57 = 10
T69 = T67 < T57
If False T69 goto L7
T58 = i
T59 = 2
T60 = j
T61 = T59 * T60
T62 = T58 / T61
i = T62
j = T67 + 1
j = T70
goto L6
L7:
```

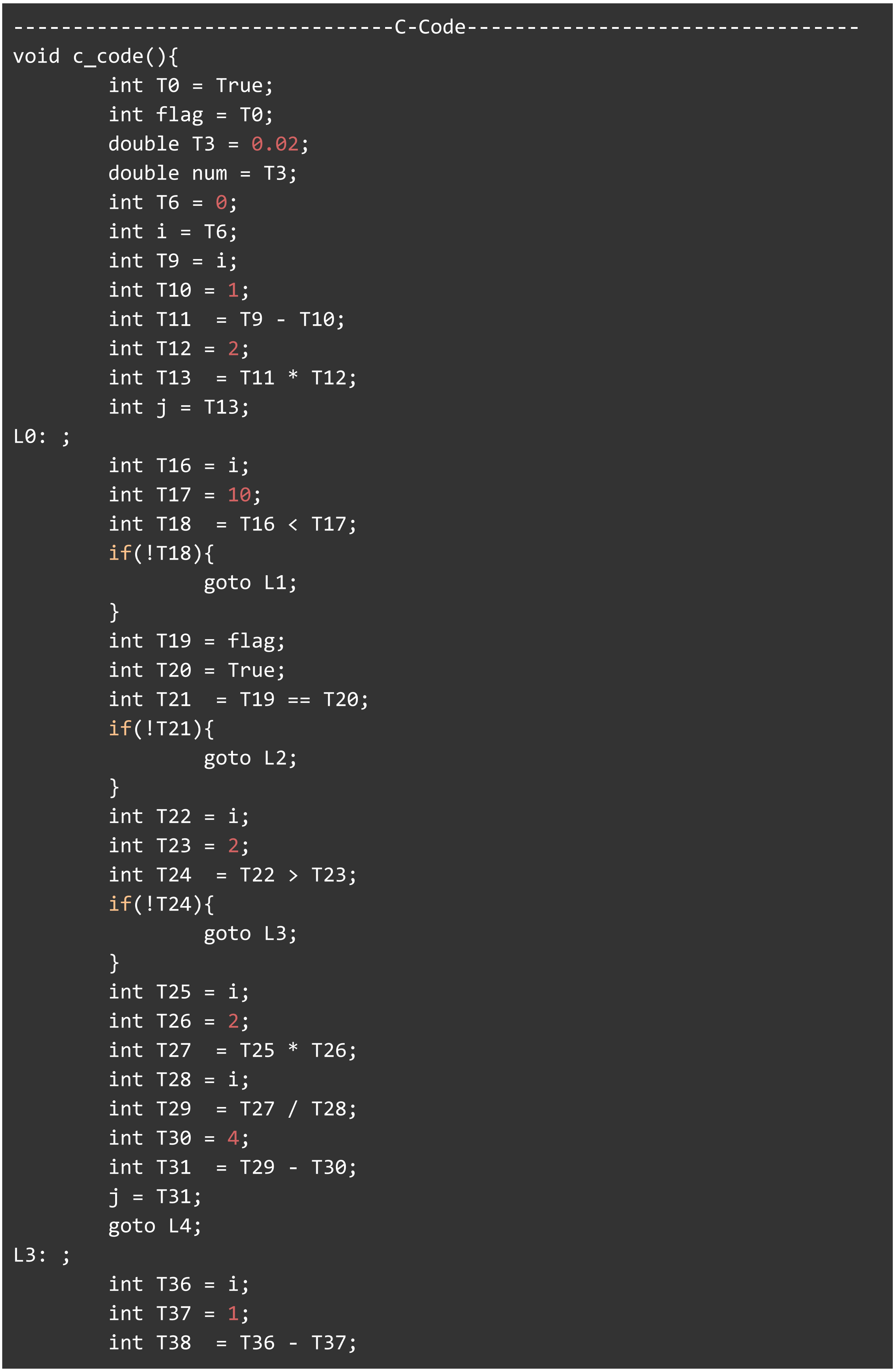


```
------------------------Symbol Tables---------------------------
Scope   Name/Value      Type            Declaration     Last Used Line
1       True            Constant                2               2
1       flag            Identifier              2               7
1       0.02            Constant                3               3
1       num             Identifier              3               14
1       0               Constant                4               4
1       i               Identifier              4               17
1       1               Constant                5               5
1       2               Constant                5               5
1       j               Identifier              5               16
1       10              Constant                6               16
1       T0              Temp
1       T3              Temp
1       T6              Temp
1       T9              Temp
1       T10             Temp
1       T11             Temp
1       T12             Temp
1       T13             Temp
1       L0              Label
1       T16             Temp
1       T17             Temp
1       T18             Temp
1       L1              Label
1       T19             Temp
1       T20             Temp
1       T21             Temp
1       L2              Label
1       T22             Temp
1       T23             Temp
1       T24             Temp
1       L3              Label
1       T25             Temp
1       T26             Temp
1       T27             Temp
1       T28             Temp
1       T29             Temp
1       T30             Temp
1       T31             Temp
1       L4              Label
1       T36             Temp
1       T37             Temp
1       T38             Temp
1       T48             Temp
1       T49             Temp
```



```
1       T50             Temp
1       T67             Temp
1       L6              Label
1       T57             Temp
1       T69             Temp
1       L7              Label
1       T58             Temp
1       T59             Temp
1       T60             Temp
1       T61             Temp
1       T62             Temp
2       True            Constant                7               7
2       j               Identifier              14              17
3       2               Constant                8               8
4       2               Constant                9               9
4       4               Constant                9               9
4       j               Identifier              9               9
16      1               Constant                11              11
16      i               Identifier              11              11
4       2               Constant                17              17
4       i               Identifier              17              17
-------------------------------------------------------------------
Remove Unused Variables
------------------------Three-address code--------------------------
0       T0      =       True
1       flag    =       T0
2       T3      =       0.02
3       num     =       T3
4       T6      =       0
5       i       =       T6
6       T9      =       i
7       T10     =       1
8       T11     -       T9      T10
9       T12     =       2
10      T13     *       T11     T12
11      j       =       T13
12      L0      Label   -
13      T16     =       i
14      T17     =       10
15      T18     <       T16     T17
16      L1      If False        T18
17      T19     =       flag
18      T20     =       True
19      T21     ==      T19     T20
20      L2      If False        T21
21      T22     =       i
```

```
22      T23     =       2
23      T24     >       T22     T23
24      L3      If False        T24
25      T25     =       i
26      T26     =       2
27      T27     *       T25     T26
28      T28     =       i
29      T29     /       T27     T28
30      T30     =       4
31      T31     -       T29     T30
32      j       =       T31
33      L4      goto    -
34      L3      Label   -
35      T36     =       i
36      T37     =       1
37      T38     -       T36     T37
38      i       =       T38
39      L4      Label   -
40      L2      Label   -
41      T48     =       num
42      T49     =       j
43      T50     *       T48     T49
44      j       =       T50
45      L0      goto    -
46      L1      Label   -
47      T67     =       0
48      j       =       T67
49      L6      Label   -
50      T67     =       j
51      T57     =       10
52      T69     <       T67     T57
53      L7      If False        T69
54      T58     =       i
55      T59     =       2
56      T60     =       j
57      T61     *       T59     T60
58      T62     /       T58     T61
59      i       =       T62
60      T67     +       T67     1
61      j       =       T67
62      L6      goto    -
63      L7      Label   -

------------------------------------------------------------------------
--
```



```
--------------------------------C-Code---------------------------------
void c_code(){
        int T0 = True;
        int flag = T0;
        double T3 = 0.02;
        double num = T3;
        int T6 = 0;
        int i = T6;
        int T9 = i;
        int T10 = 1;
        int T11  = T9 - T10;
        int T12 = 2;
        int T13  = T11 * T12;
        int j = T13;
L0: ;
        int T16 = i;
        int T17 = 10;
        int T18  = T16 < T17;
        if(!T18){
                goto L1;
        }
        int T19 = flag;
        int T20 = True;
        int T21  = T19 == T20;
        if(!T21){
                goto L2;
        }
        int T22 = i;
        int T23 = 2;
        int T24  = T22 > T23;
        if(!T24){
                goto L3;
        }
        int T25 = i;
        int T26 = 2;
        int T27  = T25 * T26;
        int T28 = i;
        int T29  = T27 / T28;
        int T30 = 4;
        int T31  = T29 - T30;
        j = T31;
        goto L4;
L3: ;
        int T36 = i;
        int T37 = 1;
        int T38  = T36 - T37;
        i = T38;
L4: ;
L2: ;
        double T48 = num;
        int T49 = j;
        double T50  = T48 * T49;
        j = T50;
        goto L0;
L1: ;
        int T67 = 0;
        j = T67;
L6: ;
        T67 = j;
        int T57 = 10;
        int T69  = T67 < T57;
        if(!T69){
                goto L7;
        }
        int T58 = i;
        int T59 = 2;
        int T60 = j;
        int T61  = T59 * T60;
        int T62  = T58 / T61;
        i = T62;
        T67  = T67 + 1;
        j = T67;
        goto L6;
L7: ;
}

------------------------------------------------------------------------
```