

ÖREBRO UNIVERSITY

COMPILERS AND INTERPRETERS

Assignment 6

Author:

Linus Baumgärtner
lbaungaertner@culba.de

Marko Reichhart
marko.reichhart@web.de

October 20, 2022

Executing syntax trees

The function `execute` as seen in listing 1 implements a recursive execution function of the syntax tree. According to the type of the tree node the specific operation is executed to its children. If the node is a leaf node the `leaf_value` is returned. The function is executed by the statement `!exe`.

Listing 1: Recursive code execution function

```
1 int execute(TreeNode* p) {
2     if (p != 0) {
3         switch (p->type) {
4             case ' ':
5                 break;
6             case NUM:
7                 return p->leaf_value;
8             case ID:
9                 if (symtable[p->leaf_value].value_initialized){
10                     return symtable[p->leaf_value].value;
11                 }
12                 else
13                     printf("Error: variable %s not initialized!", symtable[p->leaf_value].lexeme);
14                 break;
15             case '+':
16                 return execute(p->args[0]) + execute(p->args[1]);
17             case '-':
18                 return execute(p->args[0]) - execute(p->args[1]);
19             case '*':
20                 return execute(p->args[0]) * execute(p->args[1]);
21             case '/':
22                 return execute(p->args[0]) / execute(p->args[1]);
23             case '%':
24                 return execute(p->args[0]) % execute(p->args[1]);
25             case '^':
26                 return pow(execute(p->args[0]), execute(p->args[1]));
27             case '&':
28                 return execute(p->args[0]) & execute(p->args[1]);
29             case '|':
30                 return execute(p->args[0]) | execute(p->args[1]);
31             case '<':
32                 return execute(p->args[0]) < execute(p->args[1]);
33             case '>':
34                 return execute(p->args[0]) > execute(p->args[1]);
35             case '=':
36                 {
37                     symtable[p->args[0]->leaf_value].value = execute(p->args[1]);
38                     symtable[p->args[0]->leaf_value].value_initialized = true;
39                     break;
40                 }
41             case '?':
42                 return execute(p->args[0]) ? execute(p->args[1]) : execute(p->args[2]);
43             case IF:
44                 {
45                     int condition = execute(p->args[0]);
46                     if (condition)
47                         execute(p->args[1]);
48                     else
49                         execute(p->args[2]);
50                     break;
51                 }
52             case WHILE:
53                 while (execute(p->args[0]))
54                     execute(p->args[1]);
```

```

55         break;
56     case PRINT:
57         printf("%d\n", execute(p->args[0]));
58         break;
59     case READ:
60     {
61         int n;
62         std::cin >> n;
63         symtable[p->args[0]->leaf_value].value = n;
64         symtable[p->args[0]->leaf_value].value_initialized = true;
65         break;
66     }
67     case ';':
68         execute(p->args[0]);
69         if (p->args[1] != 0)
70             execute(p->args[1]);
71         break;
72     }
73 }
74 return 0;
75 }

```