PLC Laboratory 9

Obsah

Compiler of Arithmetic Expressions Using ANTLR

Extend the interpreter of arithmetic expressions from previous Laboratory 8. In this laboratory, generate a stack-based target code defining the computation.

The language description is still valid. Expressions contain +, -, *, /, % operators (with common priorities and left associativity) and parentheses. To simplify the task, consider we have only binary operators. There are no unary operators in our language. We have variables. Their identifiers compose from letters, and they have two types: `float` and `int`. Before variables are used, they need to be declared (use the same syntax as in C). Initially, they have value 0. The language is extended with = operator (lowest priority and right associativity).

Our expressions have a type. If it contains only integer numbers and variables its type is `int`. In all other cases it is `float`. In other words, if there is a floating-point number in our expression, all other integer values will be converted to float and the resulting type is `float`. In our language we also have a modulo: `%` that works only for integers.

You can start with following C# codes from previous laboratory: PLC_Lab9.zip (http://linedu.vsb.cz/~beh01/wiki_data/PLC_Lab9.zip)

The target stack-based code will consist of following instructions:

- `ADD, SUB, MUL, DIV, MOD` - it takes two values from the stack, computes the appropriate values, and stores the result on stack. If necessary, integers are automatically casted into floating-point numbers. `MOD` works only for integers.
- `PUSH (I|F) n` - it stores the value n on stack. It will be either `int` or `float`.
- `PRINT (I|F)` - it takes a value from stack and **based on its type**, prints it on output. Information about the resulting type can be obtained from **previous**

step **- type checking.**

- `LOAD a` - it loads the value of variable a to the stack.
- `SAVE (I|F) a` - it takes the value from the stack, cast it into desired variable type (given as a second parameter I|F) and stores in into variable a. Again, the type of the variable a can be taken from symbol table from type checking.

Input specification

In the input, there are expressions, they are written in the free formatting. Each expression ends with semicolon as described before.

Output specification

For each expression write the instructions (one per line) that once performed, computes the resulting value for the expression and prints the results on the screen. If there is any error in the input, you can stop the computation.

Example

- Input

```
int a,b;
a = b = 15;
a + b;
a % b;
float c;
c = a + b;
c + a;
c = a;
c + 1.1;
10 % a;
```

- Output

```
PUSH I 0
SAVE I a
PUSH I 0
SAVE I b
PUSH I 15
SAVE I b
LOAD b
SAVE I a
LOAD a
PRINT I
LOAD a
LOAD b
ADD
PRINT I
LOAD a
LOAD b
MOD
PRINT I
PUSH F 0
SAVE F c
LOAD a
LOAD b
ADD
SAVE F c
LOAD c
PRINT F
LOAD c
LOAD a
ADD
PRINT F
LOAD a
SAVE F c
LOAD c
PRINT F
LOAD c
PUSH F 1.1
ADD
PRINT F
PUSH I 10
LOAD a
```

```
MOD
PRINT I
```

Solution

- You can download the solution: PLC_Lab9_solution.zip (http://linedu.vsb.cz/~beh01/wiki_data/PLC_Lab9_solution.zip)

Citováno z „http://behalek.cs.vsb.cz/wiki/index.php?title=PLC_Laboratory_9&oldid=3813“

Stránka byla naposledy editována 17. 4. 2024 v 07:48.