Computing Arithmetic Expressions

May 10, 2023

1 Project description

Write a program that accepts an arithmetic expression written in prefix (Polish) notation, builds an expression tree, and then traverses the tree to evaluate the expression. The evaluation should start after a complete expression has been entered.

The following are some technical details.

- The expression will be read from a file.
- You will build a binary tree (the syntax tree of the expression). Each node of the tree will have a field *token* of type string, corresponding to the token (symbol or number operation) just read.
- The bulk of the work consists in constructing the tree. One possible way to construct it work like this: you will be using a stack of nodes. Given the last added node, a field *next*, with possible values $left \rightarrow right \rightarrow nil$ will indicate which pointer needs to be completed. When next becomes nil, the current node is popped off the stack and the next field of the node on the top of the stack is advanced. If it is nil it is popped as well, and so on.
  - If the current symbol is an operation symbol, a new node is created (then pushed on the stack), with the operation symbol as the value of its token field, and its next field is set to left.
  - If the current symbol is a number , a new node is created (then pushed on the stack), with the number as the value of the token field, and its next field is set to left.
  - Each node of the syntax tree may also have one field, a double called *value*. The value of the expression will be computed after the tree is created, and it will be the value of the root node.
  - To compute the value of the expression you should traverse the tree in postorder.

2 What to turn in

You will turn in

1. *a short written report* containing:
   - A description of the significant choices/issues in the design of your code.
   - One or more examples. A sample one is given as (* (- 3 1) (+ 2 5)) with value 14.
2. The source code of your program.

You may turn in the document via moodle.unibuc.ro (Class moodle).

3 Coding standards

A percentage of your grade will be based on the quality of your code, so pay attention to it. Discuss changes (if any) you made to programs presented in class. Take extra care in documenting the code you are implementing on your own. Properly modularize the code (for instance implement separate functions for significant parts of the program).

4 Deadline

June 2, 2023, noon local time. This is a strict deadline. No credit will be given for homework turned in late.