Algorithmen und Datenstrukturen
SS 2017
Blatt 6

Antonio Rodriguez-Sanchez, Sebastian Stabinger, Simon Hangl

2017–05–11

Exercise 1 : Euler tour

Extend your unsorted binary tree solution from exercise sheet 6 to hold arbitrary data (*Hint: void** in C, *Object* in Java). Write a function implementing binary Euler tour traversal, where the pre-visit, in-visit and post-visit (see lecture slides) are handled by functions that are passed as parameters in the function call.

What could this be useful for?

Exercise 2 : Expressions

Let expressions be either variables, constants, binary operations, or combinations of those. For instance, $(1 + (2 * x))$ consists of the constants 1 and 2, the variable x and the binary operations $+$ and $*$.

The structure of an expression can be modeled by the tree in exercise 1, by making constants and variables leaf nodes and binary operations inner nodes, where the operands are the corresponding child nodes.

For our exercise we only consider expressions with at most one variable, e.g. x.

- Implement a data structure Expr and the following operations:
  - Expr var() creating an expression representing the variable x
  - Expr num(int i) creating a constant representing the value i
  - Expr add(Expr a, Expr b) creating an expression representing the sum of the two given expressions
  - Expr sub(Expr a, Expr b) creating an expression representing the difference of the two given expressions
  - Expr mul(Expr a, Expr b) creating an expression representing the product of the two given expressions

- Implement the following operations for your Expr datastructure:
  - void print(Expr e) printing the expression e to the screen, e.g. $(1 + (2 * x))$. *Hint:* Is there a connection to exercise 1?
  - int eval(Expr e, int x) evaluating the expression e, assuming the value x for the variable x
- Implement test cases demonstrating the proper operation of your implementation (e.g. compute $((((2 * x) * x) + (4 * x)) - 1)$).
- What is the time complexity of print and eval?

Note: Of course, the presented signatures may be adapted to fit the programming paradigm of your chosen language.