

## Exercise 5: Expression trees

Deadline: 23<sup>th</sup> April 2021

*Expression trees* are a representations of algebraic expressions arranged in a tree data structure. In such tree, leaves contain operands and internal nodes contain operators. Expression trees are mainly used for parsing, evaluating, and modifying expressions.

### Task

Define an abstract base class `expr` for representing algebraic expressions. The class should contain an abstract method `calc()` to evaluate the expression.

In the next step, design a hierarchy of classes representing the necessary elements of the expression: operands and operators. Define operands that inherit from class `expr`:

- `num` – a floating point value of type `double`,
- `var` – a variable has a name of type `string` (variables are stored in a static set of variables),
- `const` – the constant has a name of type `string`, which is associated with a fixed value of type `double` like `pi`, `e`, `phi`, etc.

The operators represent arithmetic operations (addition, subtraction, multiplication, division, exponentiation, change of sign to the opposite, etc.) and selected mathematical functions (sine, cosine, logarithm, etc.).

In derived classes, override the method `calc()` and define stream operator `<<` for writing.

Finally write a program, which reliably tests your implementation of algebraic expressions. All objects in your program should be created on the heap (use operator `new`).

Whenever you encounter any errors, ambiguities or contradictions in the program, this should be signaled by an exception.