11. Algorithmisation Practice #3

Ján Dugáček

November 21, 2018

Table of Contents

Exercises

1. Write a convenience function that gets two vectors of same size and returns a vector of pairs
2. Write a function that returns all values and positions of inflection points of a vector that represents a function
3. Create a random number generator class that keeps its own state; you can use simple multiplication and modulo to generate random numbers

Advanced Exercises

1. Create a labyrinth class that either generates or reads a labyrinth, adds a path through it if there is none and allows accessing nodes that contain a list of pointers to other nodes accessible to it
2. Create a class that parses *markdown*, holds the parsed data and allows saving it as *markdown*, TeX or HTML; you have to support only markup for words in bold and italic

Exercises #2

1. Write an object that gives access to a `easy::vector<float>` created from file name supplied in its constructor and updates the file with the changes when the object deleted
2. Write a rational number class that is saved as a fraction and supports addition, subtraction, multiplication, division and comparison with both integers and other rational numbers
3. Create a `importanceQueue` class that has a method to add a string with some importance (two arguments) and a method to remove and return the most important string

Homework

- Create a `mathvector` class that contains a fixed number of elements that can be accessed with the `[]` operator and supports `+=`, `-=`, `*=` and `/=` operations
- You have two weeks to do it
- Challenge for the Advanced: create also a `mathmatrix` class that supports common matrix operations that work with scalars and `mathvector` as in algebra