Workshop 1/Problem 1

Ed. 1.1 as of 17.04.2019

11 апреля 2019 г.

The goal of the current task is to estimate the time complexity of different multiplication algorithms experimentally.

1. Create a class called `Multiplicator`. The class allows calculating a product of two (possibly large) integer numbers by applying different multiplication algorithms.
   We are allowed to use C++ built-in multiplication for doing one-digit multiplications (eg, 3 times 5) but not any other multiplication (eg, 23 times 55). We are allowed to use C++ built-in addition (even of large numbers).
2. In `Multiplicator` class create a method that generates a random number of *k*-digits, where *k* is a parameter of the method.
3. In `Multiplicator` class create auxiliary methods needed to implement the Grade School Multiplication algorithm and the Divide-and-Conquer Multiplication algorithm, such as: a method for getting an individual digit of a number; a method which splits one *n*-number into two (*n/2*)-numbers and so on.
4. Implement the Grade School Multiplication algorithm and the Divide-and-Conquer Multiplication algorithms as (possible static) methods of Multiplicator class. The methods calculate the algorithm work time by using C++ `<ctime>` library.
5. Create a vector member for storing estimated results of experiments in milliseconds. The elements of the vector are pairs of calculated times corresponding to both algorithms. The index of a vector element means the size of input data of an algorithm: 0 for 1-digit numbers, 1 for 2-digit numbers and so on.
6. Create a method that performs calculation of a series of numbers containing from 1 to *k* digits by applying both algorithms. Every algorithm is applied 3 times and the average time value is stored in the vector.
7. Create a method that outputs the experimental results stored in the vector as a CSV (text) file containing exactly 3 columns: size of input data and resulting time for both algorithms.
   For example:

1	12.4	15.1
2	18.2	19.3
3	25.1	40.5

8. Create a method that joins all the developed methods together and conducts experiments. The results are output to a CSV file. Based on the produced file, two line charts are drawn by using Python's Matplotlib.

Update as of 13.04.2019

9. In order to be able to represent an integer with a large count of digits it is recommended to create a custom class called Number. The internal representation of a number can be different. One of the possible representations is using `std::string` object where an individual digit is represented as a character of a string.

   Such a class can be equipped with all necessary semantics that encapsulate referencing to individual components of a number (such as a digit), performing primitive operations (such as splitting, adding two numbers using `operator+` and so on).

   How good the design of this class is can be evaluated by substituting its particular implementation (of the class Number) by another one (e.g., using `std::vector` for storing digits). This way all use cases with such a class must be preserved and no interface using this class should be adapted.

—