Workshops ADS-1, 2

16 января 2020 г.

Ver. 1.1 as of 16.01.2020

The goal of the current task is to master the previously developed *Employee* problem as implemented by using the pointer-based approach.

1. Define a structure named `Employee` describing an employee. Fields are:
   a. Name.
   b. Age.
   c. Array of wages.
2. Create a structure called `EmployeeRegister`. The class stores a collection of `Employee`s using `std::map<string, Employee*>`. The map is used for fast access to an employee by his/her name (name duplicates are not allowed).
   Mind that now instead of storing Employee objects themselves we store only pointers to `Employee` objects; the latter are placed dynamically in the heap by using the `new` operator.
3. Create a factory method called `EmployeeRegister::createNewEmployee()` which *creates* and *adds* a new `Employee` object and initializes it with given parameters: name, age, list of wages (given as a vector of doubles).
   Consider an *appropriate return value*.
4. Create a special method called `freeAll()` for the class `EmployeeRegister`. The method must properly free the memory assigned for `Employee` objects.
   - Then, consider implementation of the *destructor* of `EmployeeRegister` class, working together with this method.
5. Create a method called `findEmployee()` that obtains a name as a parameter and looks up for an employee with the same name. If an employee is found it returns a pointer to the corresponding object, otherwise `nullptr`.
   a. Implement the method in two ways. The first implementation iterates over all the objects into the internal map structure and looks for an appropriate object. The second implementation uses map features (fast key lookup) for getting access to the Employee object.
6. Create a method called `exportEmployeesAsArray()` which creates a C-style array of pointers to `Employee` and fills it up with objects from the internal map.