Лабораторная работа 4

В этой лабораторной будем учиться использовать классы `std::set`, `std::map`, `std::unordered_set`, `std::unordered_map` и другие контейнеры. Для выполнения понадобится датасет *employees.csv,* его можно найти в репозитории. В этот раз мы не будем указывать вам прототипы функций, вместо этого интерфейс функций будет описан словами.

Часть I. Функционал по работе с данными.

1. **Функция считывания данных из файла.** На вход подается название файла, на выходе - структура данных, содержащая всю информацию из файла. Предлагается хранить данные о работнике в созданной структуре `TEmployee`. Каждому работнику дополнительно нужно задать уникальный идентификатор (ID), и структура, возвращаемая функцией, должна быть отображением `Id` -> `TEmployee`.

   Идентификатор должен быть приватным: его легко получить, сложно потерять и невозможно изменить.

   Помимо этого, необходимо учесть, что эта функция может быть вызвана несколько раз в ходе работы программы и не должно возникнуть ситуации, когда 2 рабочих имеют одинаковые `Id`.
2. **Функции добавления/удаления сотрудников.** Первый параметр - структура со всеми сотрудниками, второй зависит от реализуемой функции:

   `Id` пользователя, которого нужно удалить (в случае функции удаления)

   данные о сотруднике, которого нужно добавить (объект `TEmployee`, для функции добавления)

   Функции должны корректно обрабатывать попытки удалить

несуществующего человека или добавить существующего. Новые пользователи должны иметь новые идентификаторы.

3. **Функция, генерирующая случайного сотрудника.** Простой вариант: функция принимает на вход количество работников, которое нужно сгенерировать, генерирует несколько строк для каждого (имя, фамилию, должность, департамент - последние два очень короткие, 2-3 символа), а также число из некоторого интервала - зарплату сотрудника.

   Вариант для героев: создается класс `TEmployeeGenerator`, конструктор которого принимает на вход структуру со считанными ранее из файла сотрудниками. Этот класс имеет единственный публичный метод: сгенерировать N работников. Процесс генерации работника заключается в выборе случайного имени, фамилии, должности и департамента из уже представленных в данных. Зарплата выбирается случайно в интервале между наименьшей и наибольшей зарплатами среди сотрудников, использованных в конструкторе.
4. **Функция поиска сотрудников по должности и департаменту**. Возвращает **множество** Id'шников, удовлетворяющих критериям поиска. В запросе может быть пропущен, например, департамент, тогда поиск осуществляется только по должности.
5. **Функция поиска по имени.** В запросе задается строка, в ответе - все пользователи, у которых имя **или** фамилия начинаются с этой строки.
6. **Фильтр по зарплате.** Возвращает **множество** Id'шников людей, зарплаты кото**н**рых лежат в заданном интервале. Интервал может не иметь нижней или верхней границы.
7. **Сортировка множества сотрудников по зарплате.**
   На входе - структура `Id -> TEmployee`. На выходе - вектор пар `(Id, Salary)`, отсортированный по убыванию зарплат.
8. `operator<<` **для** `TEmployee`**.**

Часть II. Другие полезные функции.

1. **Объединение/пересечение/разность двух множеств.** На входе - два множества **произвольного типа.** На выходе - одно множество того же типа, являющееся объединением/пересечением/разностью заданных.
2. `operator<<` для произвольных векторов (`std::vector`) и пар (`std::pair`).

Часть III. Аналитика данных. Каждое задание - отдельная функция.

1. Для каждого департамента посчитать минимальную и максимальную зарплату. Результат вывести в виде отображения `Dept -> pair(min, max)`.
2. Для каждого департамента посчитать среднюю зарплату. Результат вывести в виде вектора пар `pair(Dept, avgSalary)`. Вектор нужно отсортировать в порядке убывания зарплат.
3. Для каждого департамента посчитать общие годовые траты. Результаты вернуть в виде очереди с приоритетом (`std::priority_queue`).
4. Посчитать на каких должностях наибольшее число сотрудников. Вернуть в виде вектора `pair(Position, Count)` из N элементов (N - аргумент функции), отсортированного по убыванию частот.
5. Вернуть **множество** фамилий самых высокооплачиваемых сотрудников указанного департамента.

Часть IV. Тестирование.

1. Функция, которая случайно добавляет/удаляет сгенерированных сотрудников N раз (N - довольно большое, например, 1000000). Проверить, что все Id уникальны.