Школа бэкенд-разработки 2022 (осень)

4 сен 2022, 15:11:01
старт: 4 сен 2022, 15:09:20
финиш: 4 сен 2022, 20:09:20
до финиша: 04:58:16
начало: 29 авг 2022, 19:21:41
длительность: 05:00:00

C. Корпоративные закупки

	Все языки	GNU C++20 10.2
Ограничение времени	2 секунды	1 секунда
Ограничение памяти	512Mb	512Mb
Ввод	стандартный ввод или input.txt	
Вывод	стандартный вывод или output.txt	

Стартап Алисы Селезневой и Зелибобы привлек к себе внимание крупных инвесторов. Часть полученных от инвесторов денег было решено потратить на обновление офиса — новая мебель, оргтехника и другие прикольные штуки.
Алиса и Зелибоба выдвинули 5 критериев — товар должен удовлетворять всем данным критериям, чтобы его закупили в обновленный офис.

- «Наименование товара содержит подстроку **в любом регистре**» (критерий ‘NAME_CONTAINS’);
- «Цена **больше либо равна** чем» (критерий ‘PRICE_GREATER_THAN’);
- «Цена **меньше либо равна** чем» (критерий ‘PRICE_LESS_THAN’);
- «Товар поступил в продажу **не позднее** чем» (критерий ‘DATE_BEFORE’);
- «Товар поступил в продажу **не ранее** чем» (критерий ‘DATE_AFTER’);

Закупаться было решено в Выньдекс.Рынке. Для такого крупного клиента Выньдекс.Рынок предоставил стартапу персонального менеджера — да-да, именно вас.

Первым делом вам необходимо из имеющегося списка товаров на складе выбрать все товары, удовлетворяющие выданным Алисой и Зелибобой критериям.

Формат ввода

Общее описание формата входных данных:
Первая строка входных данных содержит список всех имеющихся на складе Выньдекс.Рынка товаров в формате JSON.

Следующие 5 строк имеют вид $q_i v_i$ — фильтр и соответствующее ему актуальное значение.

Подробное описание формата списка товаров

Гарантии по формату JSON:

- нет запятых после последнего элемента массива;
- все имена полей и строки обернуты в двойные кавычки.

Обозначим количество товаров в списке через N. Гарантируется, что $0 \le N \le 1000$.

Каждый товар в списке содержит следующую информацию (порядок полей не является фиксированным):

- целое число id ($0 \le id \le 2^{31} - 1$) — уникальный идентификатор. Гарантируется, что идентификаторы всех товаров попарно различны;
- строка $name$ ($1 \le |name| \le 100$) — наименование. Гарантируется, что наименование содержит только строчные и заглавные латинские буквы, а так же пробел;
- целое число $price$ ($0 \le price \le 2^{31} - 1$) — цена;
- строка $date$ в формате «dd.MM.yyyy» ($01.01.1970 \le date \le 31.12.2070$) — дата поступления в продажу.

Подробное описание формата фильтров

Гарантируется, что:

- все q_i различны между собой;
- q_i является строкой из множества (NAME_CONTAINS, PRICE_GREATER_THAN, PRICE_LESS_THAN, DATE_BEFORE, DATE_AFTER);
- в фильтре ‘NAME_CONTAINS’ v_i представляет из себя строку ($1 \le |v_i| \le 100$), содержащую только строчные и заглавные латинские буквы;
- в фильтрах ‘PRICE_GREATER_THAN’ и ‘PRICE_LESS_THAN’ v_i представляет из себя целое число ($0 \le v_i \le 2^{31} - 1$);
- в фильтрах ‘DATE_BEFORE’ и ‘DATE_AFTER’ v_i представляет из себя строку в формате «dd.MM.yyyy» ($01.01.1970 \le v_i \le 31.12.2070$).

Формат вывода

Выведите в формате JSON список товаров, удовлетворяющих всем указанным во входных данных критериям. Каждый товар должен быть выведен ровно один раз в отсортированном по возрастанию id порядке.
Выводить JSON допустимо как с дополнительными отступами и переводами строк, так и в одну строку.

Имена полей необходимо выводить в двойных кавычках.

Допустимо выводить запятую после последнего поля объекта или последнего элемента массива.

Каждый товар должен содержать информацию, аналогичную информации из входных данных:

- целое число id — уникальный идентификатор;
- строка $name$ — наименование;
- целое число $price$ — цена;
- строка $date$ в формате «dd.MM.yyyy» — дата поступления в продажу.

Пример

Ввод

```
[{"id": 1, "name": "Asus notebook","price": 1564,"date": "23.09.2021"},{"price": 2500, "id": 3, "date": "05.06.2020", "name
PRICE_LESS_THAN 2400
DATE_AFTER 23.09.2021
NAME_CONTAINS pods
PRICE_GREATER_THAN 2200
DATE_BEFORE 02.01.2022
```

Примечания

При написании решения на Java можно выбрать комплятор «Java 8 + json-simple». В этом случае вы сможете воспользоваться библиотекой json-simple для парсинга и сериализации JSON.
При написании решения на C++ можно подключить `#include "json.hpp"` для использования библиотеки json для парсинга и сериализации JSON.

Рассмотрим тестовый пример.

В нем представлено 5 товаров:

- `"id": 1, "name": "Asus notebook"price": 1564, "date": "23.09.2021"`
- `"id": 2, "name": "EaRPoDs"price": 2200, "date": "01.01.2022"`
- `"id": 3, "name": "Keyboardpods"price": 2500, "date": "05.06.2020"`
- `"id": 4, "name": "Dell notebook"price": 2300, "date": "23.09.2021"`
- `"id": 5, "name": "Airpods"price": 2300, "date": "23.09.2021"`

и следующие критерии:

- название включает подстроку `pods` в любом регистре;
- цена находится в промежутке $2200 \leq price \leq 2400$;
- дата поступления в продажу находится в промежутке $23.09.2021 \leq date \leq 02.01.2022$.

Только товары с идентификаторами 2 и 5 удовлетворяют всем критериям:

- Товар с идентификатором 1 не удовлетворяет критериям имени (нет заданной подстроки) и цены (слишком низкая);
- Товар с идентификатором 3 не удовлетворяет критериям цены (слишком высокая) и даты (слишком ранняя);
- Товар с идентификатором 4 не удовлетворяет только критериям имени (нет заданной подстроки).

Обратите внимание, что выводить необходимо товары в порядке возрастания идентификаторов (заметьте, что во входных данных товар с идентификатором 5 стоит раньше товара с идентификатором 2).

Язык C# (MS .Net 6.0)+ASP

Набрать здесь | Отправить файл

1

Отправить

Предыдущая Следующая