Дмитрий Хамутский

Школа бэкенд-разработки 2022 (лето)

6 июн 2022, 11:14:56
старт: 5 июн 2022, 17:36:08
финиш: 5 июн 2022, 22:36:08
длительность: 05:00:00
начало: 21 фев 2022, 22:47:45

C. Отчет по товарам

Ограничение времени	1 секунда
Ограничение памяти	512Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Недавно в Выньдекс.рынке среди покупателей провели опрос — какие товары они считают наиболее «интересными» для себя. На выбор предлагалось 5 различных вариантов фильтра:

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

По итогам опроса определились самые актуальные значения каждого из фильтров (по одному значению на фильтр).

Вам, как аналитику Выньдекс.Рынка, поставили задачу из имеющегося списка товаров выбрать все товары, удовлетворяющие актуальным значениям всех указанных фильтров.

Формат ввода

Общее описание формата входных данных:
Первая строка входных данных содержит список 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 фильтрам. 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"},{"id": 2, "name": "Earpods", "price": 2200, "date": "10.01.202
NAME_CONTAINS notebook
PRICE_GREATER_THAN 2000
PRICE_LESS_THAN 2400
DATE_AFTER 12.09.2021
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](https://mvnrepository.com/artifact/com.googlecode.json-simple/json-simple/1.1.1) для парсинга и сериализации JSON

Язык Python 3.7 (PyPy 7.3.3)

Набрать здесь | Отправить файл

```
import json
from datetime import datetime


def lambda_date(date1, date2, swap=False):
    if swap:
        return datetime.strptime(date1, "%d.%m.%Y") >= date2
    else:
        return date2 >= datetime.strptime(date1, "%d.%m.%Y")


def lambda_price(price1, price2):
    return price1 >= price2

def correct_candidate(name, price_l, price_r, date_l, date_r, item):
    if name in item['name'].lower() \
            and lambda_date(item['date'], date_r) \
            and lambda_date(item['date'], date_l, True) \
            and lambda_price(price_r, item['price']) \
            and lambda_price(item['price'], price_l):
        return True

    return False

data = json.loads(input())

name = ''
price_l = 0
price_r = 0
date_l = datetime.now()
date_r = datetime.now()

for i in range(5):
    line = input().split()
    if line[0] == "NAME_CONTAINS":
        name = line[1].lower()
    elif line[0] == "PRICE_GREATER_THAN":
        price_l = int(line[1])
```

Отправить

Предыдущая Следующая