# Упражнения: Въведение в структури от данни

Тествайте решението в Judge: <https://judge.softuni.bg/Contests/3186/Intro-to-Data-Structures>.

## Събития в даден обхват от време

Напишете програма, която чете набори от събития във формата "{**Събитие} | дата и време**" и две дати като **a** < **b**. Програмата трябва да отпечатва всички събития в обхвата на двете дати включително (подредете ги по дати; ако има две събития с еднаква дата, подредете ги по ред на вход).

Използвайте **Ordered Multi-Dictionary**. За да го използвате, трябва да инсталирате SoftUni.Wintellect.PowerCollections от NuGet Packages.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 5  C# Course - Group II | 15-Aug-2015 14:00  Data Structures Course | 13-Aug-2015 18:00  C# Course - Group I | 15-Aug-2015 10:00  Seminar for Java Developers | 18-Aug-2015 19:00  Game Development Seminar | 15-Aug-2015 10:00  15-Aug-2015 10:00 | 1-Sep-2015 0:00 | {C# Course - Group I,Game Development Seminar} | 15-Aug-2015 10:00  {C# Course - Group II} | 15-Aug-2015 02:00  {Seminar for Java Developers} | 18-Aug-2015 07:00 |

### Прочитане на всички събития в сортиран мулти речник

1. Прочете всички събития в **подреден** **мулти речник**

A screenshot of a computer

Description automatically generated with medium confidence

1. Първоначално нулираме текущата култура (локация), за да гарантираме, че системните локални настройки да не влияят на формата за дата и час.
2. След това създаваме **подреден мулти речник** : OrderedMultiDictionary<DateTime, string>. Той съпоставя датите със събитията.
3. Накрая четем входа ред по ред и поставяме събитията от всеки ред в мулти речника.

### Намиране на събитията в даден обхват от време

Напишете код, който да вземе ефективно поддиапазон от сортиран мулти речник. Прочетете входа от конзолата и след това използвайте метода .Range(startDate, true, endDate, true):

Text

Description automatically generated

### Отпечатване на резултата

Накрая отпечатайте изхода. Използвайте метода ToString(), за да **форматирате** датата и времето:

A picture containing chart

Description automatically generated

## Дума с префикс

Trie (Prefix Tree) са изключително полезни структури от данни, които са базирани на префикса на низ. Затова може бързо да търсете ключовите на префексите, да изброявате всички записи с един префикс и т.н. Освен това изтриването е бързо, защото е много лесно.

За да използвате структурата от данни Trie, трябва да инсталирате rm.Trie от NuGet package

Graphical user interface, application

Description automatically generated

Можете да се запознаете с методите на структурата от данни Trie на: <https://github.com/rmandvikar/csharp-trie>. Добра идея е да видите примерите.

Вашата задача е да приемите **текст** и **префикс на низ**. След това трябва да:

* Отпечатате **броя** на **думите** в текста
* Отпечатате **броя** на **уникалните думи**
* Отпечатате всички **думи** разделени с ", "
* Отпечатате **думи**, започващи с дадения **префикс**, разделени с ", "
* Премахнете всички думи с **префикс** от trie
* Отпечатате **оставащите** думи разделени с ", "

**Бързината** на Trie се дължи на неговата структура от дърво.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| She is a wonderful woman  wo | 5  5  She, is, a, wonderful, woman  wonderful, woman  She, is, a |
| My conscience hath a thousand several tongues and every tongue brings in a several tale and every tale condemns me for a villain  t | 23  17  My, conscience, condemns, hath, a, and, thousand, tongue, tongues, tale, several, every, brings, in, me, for, villain  thousand, tongue, tongues, tale  My, conscience, condemns, hath, a, and, several, every, brings, in, me, for, villain |

## Търговски център

Търговски център съдържа набор от продукти. Всеки продукт има цена и производител. Вашата задача е да моделирате търговски център и да създадете структура от данни, която да съхранява продуктите. Напишете програма, която изпълнява N команди от входа (всяка команда ще се състои от един ред):

* AddProduct име;цена;производител– добавя продукт по дадено име, цена и производител. Продуктът не се засяга, ако вече съществува продуктът с еднакво име/ производител/ цена. (позволени са повтарянията). След това трябва да отпечатате командата “**Product added**”.
* DeleteProducts производите– изтрива всички продукти, които имат определен производител. Отпечатайте “**X products deleted**”, където X e броя на изтритите продукти или “**No products found**”, ако няма такива продукти.
* DeleteProducts name;producer – изтрива всички продукти, които имат определено име и производител. Отпечатайте “**X products deleted**”, където X e броя на изтритите продукти или “**No products found**”, ако няма такива продукти.
* FindProductsByName name – намира всички продукти, които имат такова име. Отпечатайте списък от продукти във формата **{name;producer;price}**, подредени по име, производител и цена. Отпечатайте всеки продукт на нов ред. Ако няма такива продукти отпечатайте “**No products found**”.
* FindProductsByProducer producer– намира всички продукти, които имат определено име. Отпечатайте списък от продукти във формата **{name;producer;price}**, подредени по име, производител и цена. Отпечатайте всеки продукт на нов ред. Ако няма такива продукти отпечатайте “**No products found**”.
* FindProductsByPriceRange fromPrice;toPrice – намира всички продукти, които имат цена по-голяма или равна на **fromPrice** и по-малка или равна на **toPrice**. Отпечатайте списък от продукти във формата **{name;producer;price}** ,подредени по име, производител и цена. Отпечатайте всеки продукт на нов ред. Ако няма такива продукти отпечатайте “**No products found**”.

All string matching operations are **case-sensetive**.

### Вход

Входните данните трябва да се четат от конзолата.

* На първия ред ще получите **N -** броя на командите.
* На следващите **N** реда ще получите командите, обяснени по-горе.

Входните данни ще бъдат валидни. Няма нужда от допълнителни проверки.

### Изход

Изходът трябва да се отпечата на конзолата.

### Бележки

* **N** ще бъде между 1 и 50 000.
* Всички низове, посочени в командите (напр. имена на продукти и производители), се състоят от букви, цифри и интервали. Низовете различават малки и главни букви.
* Цените са дадени като реални числа с до 2 цифри след десетичната запетая (напр. 133.58, 320.3 или 10)

• Символът ‘.’ се използва като десетичен разделител.

• Цените трябва да се отпечатват с точно 2 цифри след десетичната запетая (напр. 320.30 вместо 320.3).

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 17  AddProduct IdeaPad Z560;1536.50;Lenovo  AddProduct ThinkPad T410;3000;Lenovo  AddProduct VAIO Z13;4099.99;Sony  AddProduct CLS 63 AMG;200000;Mercedes  FindProductsByName CLS 63 AMG  FindProductsByName CLS 63  FindProductsByName cls 63 amg  AddProduct 320i;10000;BMW  FindProductsByName 320i  AddProduct G560;999;Lenovo  FindProductsByProducer Lenovo  DeleteProducts Lenovo  FindProductsByProducer Lenovo  FindProductsByPriceRange 100000;200000  DeleteProducts Beer;Ariana  DeleteProducts CLS 63 AMG;Mercedes  FindProductsByName CLS 63 AMG | Product added  Product added  Product added  Product added  {CLS 63 AMG;Mercedes;200000.00}  No products found  No products found  Product added  {320i;BMW;10000.00}  Product added  {G560;Lenovo;999.00}  {IdeaPad Z560;Lenovo;1536.50}  {ThinkPad T410;Lenovo;3000.00}  3 products deleted  No products found  {CLS 63 AMG;Mercedes;200000.00}  No products found  1 products deleted  No products found |

### Насоки

В дадения скелет са ви дадени всички класове, от които ще се нуждаете. Въпреки това трябва да попълните някой методи. Нека първо да видим какво имаме.

Да разгледаме класа Product в файла Product.cs. Имаме имплементирана структура на класа – всички свойства, които са ви нужни, CompareTo(product) метод за сравняване на продукти и написан метод ToString() за отпечатване на информацията.

Забележете, че в класа ProductList имаме комбинирани структури от данни за създаване на подходяща колекция за нашите данни:

Graphical user interface, text, application

Description automatically generated

Например използваме Dictionary<string, OrderedBag<Product>> за съхраняване на имената, защото искаме бързо търсене по име (затова използваме речник като имената са ключове), в същото време можем да вземем продуктите с имена без да **минаваме** през всичките продукти и да ги **сортираме**. Използването на OrderedBag<Product> е възможно заради метода CompareTo(product), който имплементирахме в класа Product.

В допълнение към свойствата, сме имплементирали някои **методи** за по-лесна работа със сложни структури от данни. **Внимателно** разгледайте методите, защото ще ги използвате.

За още по-голяма простота за използване на структури от данни и за по-чист код, сме имплементирали допълните методи за **речника** в класа DictionaryExtensions. Разгледайте ги също.

Накрая в класа StartUp имате ProductList и някои **методи**. **Имплементирайте** липсващите методи като използвате методите от ProductList, които вече разгледахте.