# Упражнения: Въведение в структури от данни

Тествайте решението в Judge: <https://judge.softuni.org/Contests/4171/12-Data-Structures-Overview-Basics>.

## Уникални имена

Напишете програма, която чете **списък от имена** на един ред, разделени със **запетая и интервал** (", "). Изполвайте **подходяща структура от данни**, за да извлечете и отпечатате само **уникалните имена**, подредени по **азбучен ред**, всяко на **нов ред**. Проверката за уникалност трябва да бъде **case-insensitive**, т.е. ако първо се въведе "**Brian**", а след това и "**brian**", като уникално се брои **само първото име**.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| Marry, Ivan, Tommy, Ivan | Ivan  Marry  Tommy |
| John, Jenny, john, Sofiya, Martin, jenny, Dan | Dan  Jenny  John  Martin  Sofiya |

### Насоки

1. Първо трябва да изберете подходяща структура от данни. В тази задача се изисква да пазите **уникални записи** и те да бъдат подредени **по азбучен ред**. Най-подходящо ще бъде да използвате **SortedSet<T>**.
2. Създайте помощен **списък**, в който да запазите **първоначалния вход**:



1. Създайте **сортирано множество** (**SortedSet**) за **уникалните** имена. За да бъде **case-insensitive**, трябва да използвате настройката **StringComparer.OrdinalIgnoreCase**:



1. Отпечатайте всяко **уникално** име на **нов ред**.

## Проверка за палиндром

Напишете програма, която чете **дума** (стринг) от конзолата и проверява дали думата е **палиндром**. Палиндром се нарича дума, която се чете по **еднакъв начин** от **ляво надясно** и от **дясно наляво**. Сравнението трябва да е **case-insensitive**.

Отпечатайте резултата в следния формат:

* Ако думата **е** **палиндром**:  
  **"The word {дума} is a palindrome."**
* Ако думата **не е палиндром**:  
  **"The word {дума} is not a palindrome."**

В резултата думата трябва да се отпечатва с **малки букви** (**lower-case**).

**Важно**: Не използвайте метода **Reverse()**. Използвайте съответната структура от данни, която ще ви предостави необходимата функционалност.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| hello | The word hello is not a palindrome. |
| racecar | The word racecar is a palindrome. |
| CivIC | The word civic is a palindrome. |

### Насоки

* За да проверим дали дадена дума е **палиндром**, можем да проверим дали **оригиналната дума** е **равна** на **обърнатата** ѝ версия. Ако са **еднакви**, думата е **палиндром** (чете се еднакво от ляво надясно и от дясно наляво).

1. Прочетете **оригиналната дума** от конзолата и я конвертирайте в **lowercase** (за да бъдат проверките ни case**-insensitive**).
2. Създайте **стек**, който първоначално да съдържа **всички символи** от **оригиналната дума**:



1. Създайте стринг за **обърнатата дума**. Може да го направите чрез създаване на **празен стринг** ("") или със **StringBuilder**:

A blue and black text

Description automatically generated

1. Тъй като използвате **стек**, може на всяка итерация, до **изчерпване на елементите** в него, да **вадите буква** и я **добавяте към обърнатия стринг**:

A close-up of a computer screen

Description automatically generated

1. След приключване на цикъла проверете дали **оригиналната** версия е **равна** на **обърнатата** и отпечатайте съответния резултат.

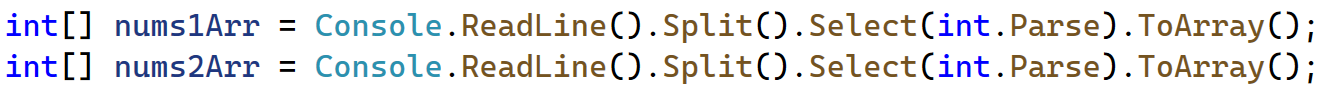
## Обединение на две множества

Създайте програма, която чете два **списъка с числа** (числата във всеки списък са **разделени с интервал**). **Премахнете повторенията** в рамките на отделните списъци. След това **обединете получените множества** в едно и отпечатайте неговите **елементи**, разделени с **интервал** (" ").

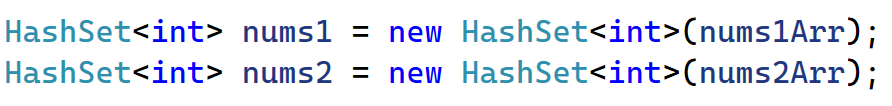
|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 1 3 -45 0 23  3 0 6 23 50 | 1 3 -45 0 23 6 50 |
| -1 3  1 3 4 | -1 3 1 4 |

### Насоки

1. Прочетете данните за двата **списъка от числа**, като ги запазите в **масиви**:



1. Тъй като трябва да останат само **уникалните числа**, създайте **множество** за всеки от двата масива:



1. Създайте **множество за обединението**, което първоначално ще бъде **празно**. Избираме отново **множество**, а не списък, защото при **обединение** може да има **повтарящи се елементи**, които трябва да се премахнат:



1. Итерирайте с цикъл през **първото множество** и добавете елементите му към **numsUnion**:

A close-up of a computer screen

Description automatically generated

1. Итерирайте по същия начин през **второто множество** и **добавете** неговите **елементи** към **numsUnion**.
2. **Отпечатайте** числата на конзолата, разделени с **интервал**.

## 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 |

### Насоки

1. Нулирайте текущата **култура** (**локация**), за да гарантирате, че системните локални настройки няма да влияят на **формата** за **дата** и **час**:

A screenshot of a computer

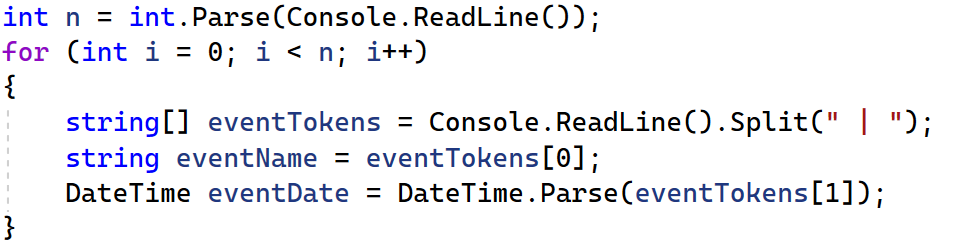
Description automatically generated with medium confidence

1. След това създайте **подреден мулти речник** : OrderedMultiDictionary<DateTime, string>. Той подрежда датите на събитията:

A screenshot of a computer

Description automatically generated with medium confidence

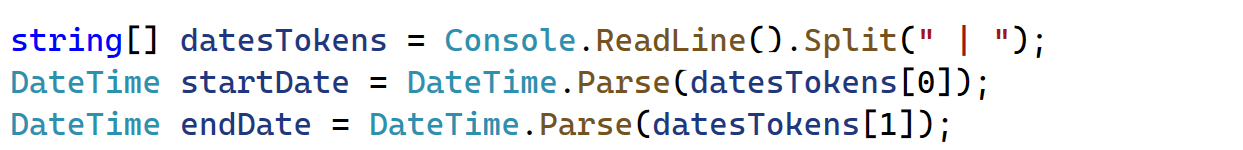
1. Прочете **входа** ред по ред:



1. Добавете **събитието** в **мулти** **речника**:



1. След това прочете **двете** **дати**:



1. Вземете датите в обхвата като използвате .Range(startDate, true, endDate, true):



1. Накрая отпечатайте датите чрез **foreach-цикъл**:

A picture containing chart

Description automatically generated

## Дума с представка

Trie (Prefix Tree) са изключително полезни структури от данни, които са базирани на представка на низ. Затова може бързо да търсете ключовите на префексите, да изброявате всички записи с един представка 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 **представка 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 **представка**, разделени с ", "
* Премахнете всички думи с **представка** от trie
* Отпечатате **оставащите** думи разделени с ", "

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

### Примери

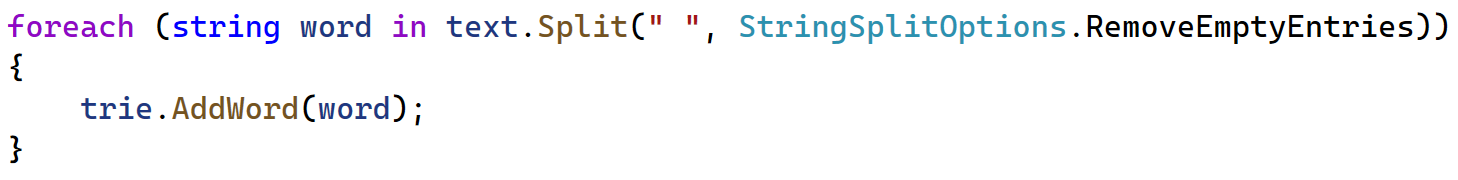
|  |  |
| --- | --- |
| **Вход** | **Изход** |
| She is a wonderful woman  wo | 5  5  She, is, a, wonderful, woman  wonderful, woman  She, is, a |
| My conscience hath a thousand several tongues and every tongue brings in a several tale and every tale condemns me for a villain  t | 23  17  My, conscience,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

### Насоки

1. Създайте **Trie**:



1. Прочетете **текста** и **представката**.
2. Създайте **foreach-цикъл**, който ще обхожда всички думи на текста:



1. В него добавете думата към **trie**:



1. Отпечатайте **броя** на **думите** в текста чрез метода **Count()**:



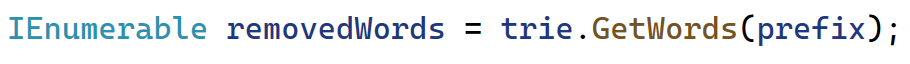
1. Отпечатайте **броя** на **уникалните думи** чрез метода **UniqueCount()**:



1. След това създайте променлива, която ще получи всички думи чрез метода **GetWords()**:



1. Отпечатайте променливата като разделите **думите** с ", ".
2. Създайте променлива, която ще получи всички думи, които започват с представката чрез метода **GetWords(представка)**:



1. Отпечатайте променливата, като раделите **думите** с ", ".
2. Премахнете всички думи, съдържащи **представка** в **trie**:



1. Отпечатате **оставащите** думи разделени с ", ":



## 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**”.

Всички операции за съвпадение на стрингове трябва да бъдат **case-sensitive**.

### 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 |