# Упражнения: Регулярни изрази

Тествайте решението в Judge: <https://judge.softuni.org/Contests/4166/08-Regular-Expressions-Basics>.

## Търсене на пълно име

Напишете програма на C#, която търси всички пълни имена от списък с имена. Накрая ги отпечатайте.

### Примери

|  |
| --- |
| **Вход** |
| Ivan Ivanov, Ivan ivanov, ivan Ivanov, IVan Ivanov, Test Testov, Ivan Ivanov |
| **Изход** |
| Ivan Ivanov  Test Testov |

### Насоки

#### Писане на регулярни изрази

Първо трябва да напишем **регулярен израз** с валидни имена по следните изисквания:

* Пълното име е валидно, когато
  + Съдържа само две думи
  + Всяка дума започва с главна буква
  + След първата буква всички останали букви в думата са малки
  + Всяка дума трябва да има минимална дължина от 2 символа.
  + Между двете букви трябва да има интервал

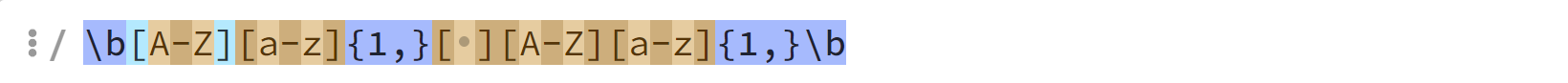
1. Използвайте онлайн сайт за тестване на регекс като [https://regex101.com](https://regex101.com/)
2. Уточнете, че всяка дума трябва да започва с главна буква от английската азбука:



1. Уточнете че търсете две думи с интервал между тях (приема се само символа **'** **',** без табове и нови редове):



1. За да предотвратите засичането на букви на нови редове, сложете "\b" в началото и края на вашия регекс:



1. С цел да проверите вашия регекс може да използвате таблицата по-долу:

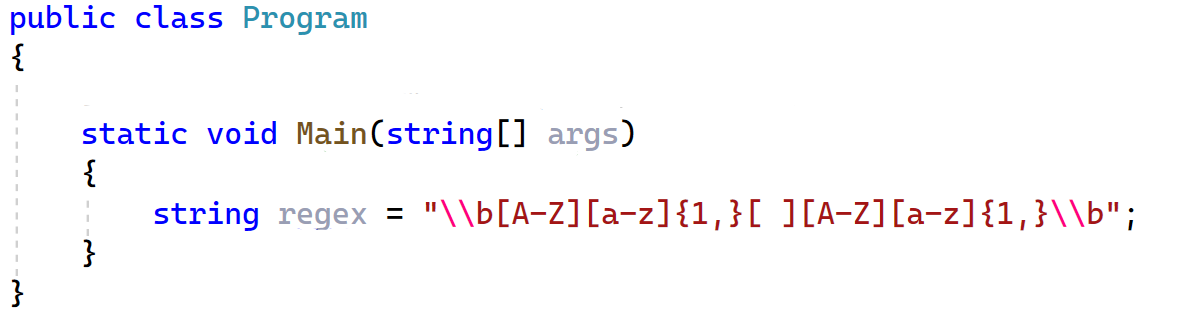
|  |  |
| --- | --- |
| **Верни изрази** | **Грешни изрази** |
| Ivan Ivanov | ivan ivanov, Ivan ivanov, ivan Ivanov, IVan Ivanov, Ivan IvAnov, Ivan Ivanov |

1. Резултатите от търсенето трябва да изглеждат така:



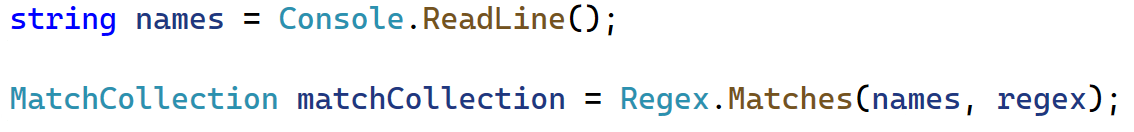
#### Имплементация на решението в C#

1. Създайте **променлива от тип низ** исложете **регулярния израз**:

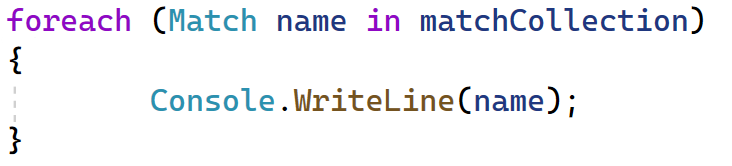


Забележка: Добра идея е да използвате символа '@' за да складирате регулярния израз, защото някои символи като '\' могат да причинят грешка.

1. **Прочетете входа** и да **извлечете съвпаденията** като използвайте класа MatchCollection:



1. Накрая отпечатайте правилните имена чрез foreach**-цикъл**:



## Търсене на телефонни номера

Напишете регулярен израз, който да търси валидни номера от София. След като ги намерите, трябва да ги отпечатате на конзолата, разделени със запетая и интервал “, ”.

Номера е валиден, когато са изпълнени следните условия:

* Началото в низа има символа **"+359"**.
* След това има винаги **2**.
* Следващите числа са 7 наброй (разделени на две групи от 3 и 4 числа):
* Различни части трябва да бъдат разделени с интервал или тире (**'-'**).

### Примери

|  |
| --- |
| **Вход** |
| +359 2 222 2222,359-2-222-2222, +359/2/222/2222, +359-2 222 2222 +359 2-222-2222, +359-2-222-222, +359-2-222-22222 +359-2-222-2222 |
| **Изход** |
| +359 2 222 2222, +359-2-222-2222 |

### Насоки

#### Писане на регулярни изрази

1. Уточнете че търсите **"+359"** в началото:

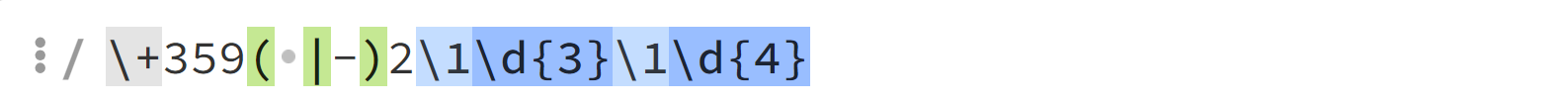


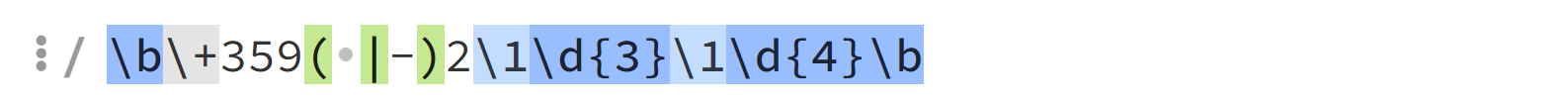
1. След това създайте група, която проверява дали символът **"+359"** и **"2"** са разделени с **интервал** или **тире**:



1. Уточнете, че търсите **3** числа:



1. Използвайте обратна референция към първата група:  
   
2. Отново уточнете, че търсите 4 числа:  
   
3. Накрая добавете "\b" в началото и края на вашия регекс:



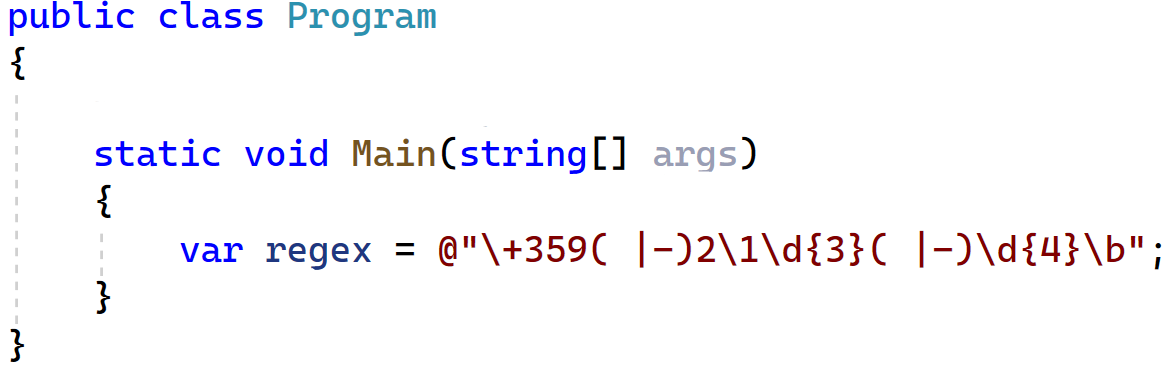
1. Можете да използвате таблицата по-долу за да си проверите регекса:

|  |  |
| --- | --- |
| **Верни изрази** | **Грешни изрази** |
| +359 2 222 2222  +359-2-222-2222 | 359-2-222-2222, +359/2/222/2222, +359-2 222 2222  +359 2-222-2222, +359-2-222-222, +359-2-222-22222 |

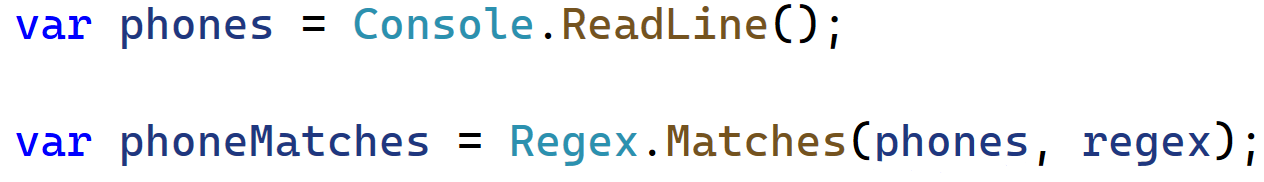
#### Имплементация на решението в C#

Сега е време да напишем решението, затова да запретваме ръкави!

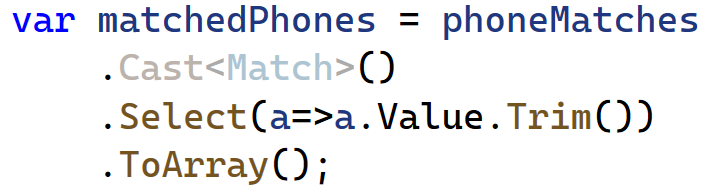
1. Създайте **променлива** както в предишната задача:



1. **Прочетете входа** и да **извлечете съвпаденията** като използвайте класа MatchCollection:



1. Опитайте се да **отпечатате** **всички** **съвпадения** на един ред. Използвайте LINQ понеже MatchCollection връща колекция:
2. Превърнете всеки един елемент от MatchCollection в тип Match чрез **метода** Cast<Match>().
3. Тъй като всеки елемент е Match, можете да вземете стойностите на всяко съвпадение чрез метода Select(). Може да използвате метода Trim() за стойностите, за да премахнете всички начални или крайни интервали.
4. След като вземете стойностите може да използвате ToArray(), за да превърнете колекцията в масив.
5. Ето така изглежда нашата LINQ заявка:



1. След това отпечатваме валидните номера чрез string.Join():



## Търсене на дати

Напишете програма, която търси дати във формата “dd{разделител}MMM{разделител}yyyy”. Използвайте **групи** в регулярния израз.

### Писане на регулярни изрази

Датата е валидна, когато са изпълнени следните условия:

* Винаги започва с **две числа**, последвани от **разделител**.
* Има **една главна** и след това **две малки** букви. (Jan, Mar).
* След това има **разделител**, последван от точно **четири цифри** (за годината).
* Разделителят може да бъде **три типа**: точка (“.”), тире (“-“) и наклонена черта (“/”).
* Разделителят трябва да е еднакъв за цялата дата, например дата 13.03.2016 е валиден но, 13.03/2016 не е валиден. Използвайте **обратна референция,** за да проверите това.

### Примери

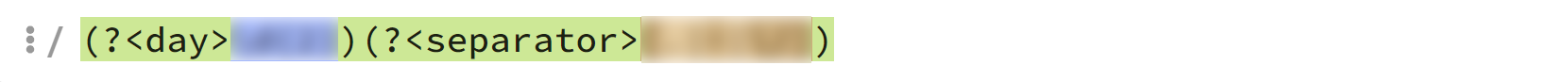
|  |
| --- |
| **Вход** |
| 13/Jul/1928, 10-Nov-1934, , 01/Jan-1951,f 25.Dec.1937 23/09/1973, 1/Feb/2016 |
| **Изход** |
| Day: 13, Month: Jul, Year: 1928  Day: 10, Month: Nov, Year: 1934  Day: 25, Month: Dec, Year: 1937 |

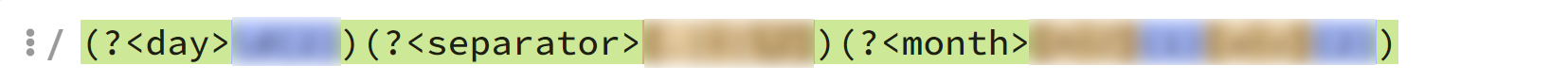
### Насоки

#### Писане на регулярни изрази

1. Създайте **група** за **деня**, която търси **две цифри**.
2. Добавете **име на групата**, за да я търсите по-лесно:



1. След това добавете разделителя (**тире**, **точка** или **наклонена черта**) като група с име. Използвайте символът **"\"** преди наклонената черта: 
2. Създайте група за месеца. Месецът съдържа главна латинска буква и две малки латински букви. Наименувайте я **"month"**:



1. След това добавете разделителя чрез обратна референция.
2. Добавете група за **годината** с име, която се състой от **четири цифри**:

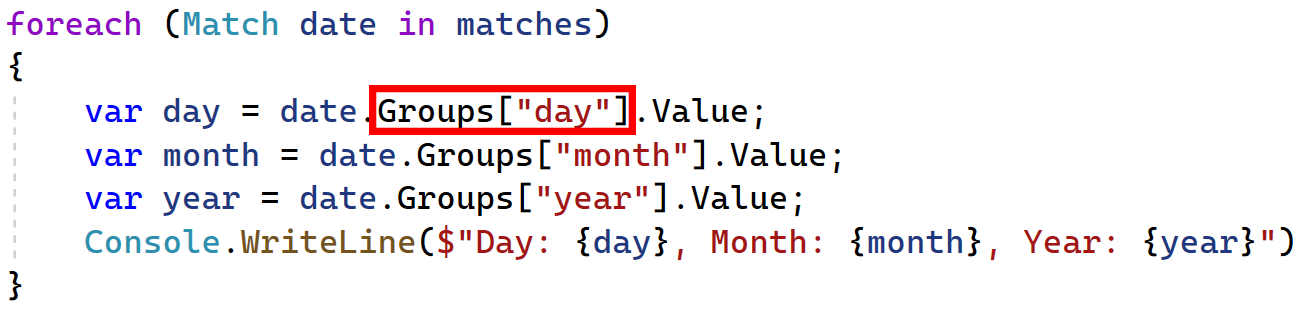


1. Накрая добавете "\b" в началото и края на вашия регекс.
2. Може да следвате таблицата по-долу за регекса:

|  |  |
| --- | --- |
| **Верни изрази** | **Грешни изрази** |
| 13/Jul/1928, 10-Nov-1934, 25.Dec.1937 | 01/Jan-1951, 23/sept/1973, 1/Feb/2016 |

#### Имплементация на решението в C#

1. **Прочетете входа** и да **извлечете съвпаденията** като използвайте класа MatchCollection.
2. След това минете през всеки един Match и извлечете деня, месеца и годината от групите. Можете да използвате специален синтаксис в C#, който да вземе стойността на определена група, както взимате стойностите от речник:



## Мебели

Напишете програма, която пресмята цената на всички мебели. Ще получавате предмети докато не получите **"Purchase"**. Всеки ред ще бъде в следния формат:

**">>{мебел}<<{цена}!{качество}"**

Цената може да бъде цяло число или число с плаваща запетая. Отпечатайте мебелите и общата цена. На всеки ред трябва да отпечатате предеметите в следния формат:

**"Bought furniture:**

**{мебел1}**

**{мебел2}**

**…"**

Накрая трябва да отпечатате следното съобщение: "Total money spent: {общата цена}" **-** форматирайте сумата до второто число след десетичната запетая.

Ако получите невалиден вход, пропуснете командата.

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| >>Sofa<<312.23!3  >>TV<<300!5  >Invalid<<!5  Purchase | Bought furniture:  Sofa  TV  Total money spent: 2436.69 | Само **Sofa и TV** са валидни. Умножаме цените им по количеството и отпечатваме разултата |

## Приходите на бара в СофтУни

Нека да си вземем почивка и да отидем в бара на СофтУни. Време е хората от бара да се прибират по домовете си. Вие сте човекът, който трябва да пресметни оборота за деня. Докато не получите "end of shift", ще получавате редове от команди, но преди да обработите редовете, ще трябва да направите няколко проверки.

Всеки правилен вход трябва да има **име на** **клиент**, **продукт**, **количество** и **цена**:

* Ако името на клиента е **обградено с** **"%"** и започва с главна буква, следвана от малки, то той е валиден
* Ако един продукт **съдържа букви** и е **обграден с** **"<"** и **">"**, то той е валиден
* Ако количеството е **число**, **обграден с "|"**, то то е валидно
* Ако цената е **реално число**, **последвано с** **"$"**, то тя е валидна

Всеки валиден ред трябва да бъде в реда: **име на клиент, продукт, количество и цена**.

Между всяка част може да има и други символи, с изключение на (**'|', '$', '%' и '.'**)

За всеки валиден ред трябва да от печатате на конзолата:

"{име на клиент}: {продукт} - {оборот}"

Когато получите "end of shift", отпечатайте оборота за деня, закръглен до втория знак след десетичната запетая във формата: "Total income: {оборот за деня}"**.**

### Вход

* Ще получавате редове, докато не получите "end of shift".

### Изход

* За всеки валиден ред трябва да от печатате на конзолата:

"{име на клиент}: {продукт} - {оборот}"

* Когато получите "end of shift", отпечатайте оборота за деня, закръглен до втория знак след десетичната запетая във формата: "Total income: {оборот за деня}"**.**

### **Примери**

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| %George%<Croissant>|2|10.3$  %Peter%<Gum>|1|1.3$  %Maria%<Cola>|1|2.4$  end of shift | George: Croissant - 20.60  Peter: Gum - 1.30  Maria: Cola - 2.40  Total income: 24.30 | Всяка команда е правилна, затова пресмятаме цената на всеки продукт.  Накрая отпечатваме оборота за деня |
| %InvalidName%<Croissant>|2|10.3$  %Peter%<Gum>1.3$  %Maria%<Cola>|1|2.4  %Valid%<Valid>valid|10|valid20$  end of shift | Valid: Valid - 200.00  Total income: 200.00 | На първия ред имаме невалидно потребителско име, затова пропускаме реда. На следващият ред няма количество за продукта. На третият ред няма валидна цена. Само на четвърти ред всичко е правилно. |