# 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.

### 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

За да ви помогнем, ще ви покажем няколко стъпки:

1. Използвайте онлайн сайт за тестване на регекс като [https://regex101.com](https://regex101.com/)
2. Вижте как да използвате набори от символи (обозначени с квадратни скоби - "[]")
3. Уточнете че търсете две думи с интервал между тях (приема се само символа **'** **',** без табове и нови редове)
4. За всяка дума уточнете, че трябва да започва с главна буква от английската азбука.
5. Останалите букви след първата трябва да бъдат малки. Ще има една или повече малки букви – използвайте правилния **quantifier**.
6. За да предотвратите засичането на букви на нови редове, сложете "\b" в началото и края на вашия регекс. .

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 |

Резултатите от търсенето трябва да изглеждат така:

Graphical user interface, application

Description automatically generated

След като сте създали регулярния израз трябва да го напишете в C#

### Имплементация на решението в C#

1. Създайте нов проект на C# и копирайте регулярния израз в **низ**:

A picture containing table

Description automatically generated

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

1. Време е да **прочетем входа** и да **извлечем съвпаденията**. Затова ще използваме класа MatchCollection

Text

Description automatically generated with medium confidence

1. След това извличаме резултатите, минаваме през **MatchCollection** и принтираме намереното

Graphical user interface, text, application

Description automatically generated

### Примери

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

## 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 “, ”.

### Писане на регулярни изрази

Номера е валиден, когато са изпълнени следните условия:

* Започва с "**+359**"
* След това трябва да има винаги 2
* Следващите числа трябва да бъдат 7 наброй (разделени на две групи от 3 и 4 числа):
* Различни части трябва да бъдат разделени с интервал или тире ('**-**').
* Използваме **quantifier-и** за да намерим специфичните числа.
* Използвайте група, за да се уверите, че разделителят е един от следните символи: интервал или тире. Не се приема комбинация от двата символа (+359 2-111 111 е невалиден, защото съдържа тире и интервал). Използвайте **обратна референция.** Добавете граница на думата в края на съвпадението, за да избегнете частични съвпадения (последният пример от дясната страна).
* Уверете се, че знака '+' е в **начало на низа**.

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 запредваме ръкави!

1. Първо трябва да създадем променлива както в предишната задача:

A picture containing text

Description automatically generated

1. Отново използваме класа MatchCollection за търсене на съвпадения

Text

Description automatically generated

1. Нека да се опитаме да отпечатаме съвпаденията само чрез един ред код. Понеже MatchCollection връща колекция, може да използваме метода LINQ.

За да вземем всички съвпадения в масив от низ трябва да изпълним няколко манипилации в MatchCollection:

1. Превръщаме всеки един елемент от MatchCollection в тип Match чрез Cast<Match>().
2. Тъй като всеки елемент е Match, можем да вземем стойностите на всяко съвпадение чрез Select(). Може да използваме метода Trim() за стойностите, за да премахнем всички начални или крайни интервали
3. След като взехме стойностите може да използваме ToArray(), за да превърнем колекцията в масив

Ето така изглежда нашата LINQ заявка:

Text

Description automatically generated with medium confidence

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

Text

Description automatically generated with medium confidence

### 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 |

## 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 с раздетел.
* Има една главна и след това две малки букви. (Jan, Mar).
* След това има разделител последван от точно четири числа (за годината).
* Разделителя може да бъде три типа: точка (“.”), тире (“-“) и наклонена черта (“/”).
* Разделителя трябва да раздели цялата дата например (13.03.2016 е валидно, 13.03/2016 е невалидно). Използвайте **обратна референция,** за да проверите това.

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 |

Използвайте **групи** за **деня**, **месеца**, и **годината.**

Тъй като задачата изисква по-сложен регекс, включва наименувани групи. Нека сега да видим как се създава:

1. Първо не искаме да има други символи преди нашата дата, затова ще използваме “\b”:  
   Graphical user interface, application

   Description automatically generated
2. След това ще търсим деня. Уточняваме, че търсим точно две цифри и тъй като по-късно ще искаме да извлечем този ден, ще го сложим в група:  
   Table

   Description automatically generated with low confidence
3. Ще добвим име на групата, за да я търсим по-лесно:  
   Table

   Description automatically generated with low confidence
4. След това идва разделителя - тире, точка или наклонена черта. Може да използваме клас от символи:  
   A picture containing table

   Description automatically generated
5. Понеже искаме да използваме същия разделите и по-късно, можем да го сложем в група:  
   A picture containing graphical user interface

   Description automatically generated
6. След това идва месеца, който съдържа главна латинска буква и точно две малки латински букви:  
   A picture containing application

   Description automatically generated
7. След това ще търсим предишния разделител. Може да използваме обратна референция за това:  
   A picture containing application

   Description automatically generated
8. Сега ще търсим годината, която се състои от точно четири цифри:  
   A picture containing application

   Description automatically generated
9. Тъй като не искаме да има други символи след датата можем отново да използваме “\b”:  
   A picture containing table

   Description automatically generated

Време е да намерим **валидните дати** и да ги отпечатаме на входа на **нов ред** във формата: “Day: {ден}, Month: {седмица}, Year: {година}”,

### Имплементация на решението в C#

1. Първо ще сложим регекса в променлива и ще получим MatchCollection:

Graphical user interface

Description automatically generated with medium confidence

1. Тъй като регекса работи различно в различните езици, преди да продължим, ще променим обратната референция от \2 **на** \1 Правим така, защото обратните референции в C# не броят наименуваните групи за обратни референции.
2. След това минаваме през всеки един Match и извличаме деня, месеца и годината от групите. Може да използваме специален синтаксис в C#, който да вземе стойностите като ключ, както взимаме стойностите от речник:

Text

Description automatically generated

### 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 |

## 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 **заобградено с** **"%"** и започва с главна буква, следвана от малки, то той е валиден
* Ако един продукт **съдържа букви** и е **заобграден с** **"<"** и **">"**, то той е валиден
* Ако количеството е **число**, **заобградено 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 |