Практикум. Создание объекта Regex для проверки текста

Упражнение 1. Выборка телефонных номеров и почтовых индексов

В этом упражнении вы напишете код, отличающий телефонные номера и почтовые индексы от недопустимых данных.

1. Скопируйте С#-версию папки Chapter03\Lessonl-Exercisel и откройте файл решения.
2. Переделайте метод IsPhone так, чтобы он возвращал true при вводе чисел в любом из следующих форматов (используйте не более одной строки кода):

* (555)555-1212
* (555) 555-1212 и 555-555-1212
* 5555551212

Пример метода IsPhone может быть таким:

// C#

static bool IsPhone(string s) {

return Regex.IsMatch(s, ^\(?\d{3}\)?[\s\-]?\d{3}\-?\d{4}$");

}

Каждый компонент этого регулярного выражения соответствует обязательной или необязательной части телефонного номера:

□ ^

— началу строки;

□ \(?

— открывающейся круглой скобке (необязательно). Круглой скобке предшеству­ет обратная косая черта, потому что круглая скобка — спецсимвол в регулярных выражениях; вопросительный знак делает круглую скобку необязательной;

□ \d{3}

— трем цифрам;

□ \)?

— закрывающейся круглой скобке (необязательно). Скобка помечена обратной косой чертой, так как круглая скобка является спецсимволом в регулярных выражениях; знак вопроса делает круглую скобку необязательной;

□ [\s\-]?

— пробелу («\s») или дефису («\-»)> отделяющему код города. Знак вопроса дела­ет пробел или дефис необязательным;

□ \d{3}

— трем цифрам;

□ \-?

— дефису (необязательно);

□ \d{4}$

— требует завершения строки четырьмя цифрами.

3. Строку кода завершает вызов IsZip, он возвращает true, если ввод имеет один из следующих форматов:

* 01111
* 01111-1111

Код IsZip выглядит примерно так:

// C#

static bool IsZip(string s) {

return Regex.IsMatch(s, @" ^\d{5}(\-\d{4})?$

}

Компоненты этого регулярного выражения соответствуют частям почтового индекса:

□ ^

— началу строки;

□ \d{5}

— пяти цифрам;

□ (\-\d{4})?

дефису, следующему после цифр (необязательно, так как это выражение нахо­дится в круглых скобках и сопровождается знаком вопроса);

□ $

— концу строки;

4. Соберите и запустите проект. Вывод должен быть таким:

(555)555-1212 is a phone number (555) 555-1212 is a phone number 555-555-1212 is a zip code 5555551212 is a zip code 01111 is a zip code 01111-1111 is a zip code 47 is unknown 111-11-1111 is unknown

Если вывод отличается, исправьте регулярное выражение.

Упражнение 2. Переформатирование строк

В этом упражнении необходимо привести телефонный номер к стандартному формату: (###) ###\_####.

1. Откройте проект, созданный в Упражнении 1.
2. Добавьте метод ReformatPhone, возвращающий строку и принимает единственный строковый параметр. Используя регулярные выражения, он должен принимать теле­фонные номера в формате, используемом в упражнении 1, и приводить их к форма­ту (###) ###-####. Метод IsZip выглядит примерно так:

// C#

static string ReformatPhone(string s) {

Match m = Regex.Match(s, @" ^\(?(\d{3})\)?[\s\-]?(\d{3})\-?(\d{4})$"); return String.Format("({0}) {1}-{2}", m.Groups[1], m.Groups[2], m.Groups[3]);

}

Обратите внимание, что это регулярное выражение почти такое же, как в методе IsPhone, и отличается только выражениями \d{n], заключенными в круглые скоб­ки. Оно разбивает строку на группы, которые легко отформатировать с помощью String.Format.

3. Измените метод Main так, чтобы в цикле foreach записывался результат ReformatPhone(s),  
а не просто s, например:

foreach (string s in input) {

if (IsPhone(s)) Console.WriteLine(ReformatPhone(s) + " is a phone number");

else

if (IsZip(s))

Console.WriteLine(s + " is a zip code");

else Console.WriteLine(s + " is unknown");

}

4. Соберите и запустите проект. Вывод должен быть таким:

(555) 555-1212 is a phone number (555) 555-1212 is a phone number (555) 555-1212 is a phone number (555) 555-1212 is a phone number 01111 is a zip code 01111-1111 is a zip code 47 is unknown 111-11-1111 is unknow:

Обратите внимание, что телефонные номера переформатируются, независимо от их исходного формата. Если полученный вывод отличается, исправьте регулярное вы­ражение.