Частное учреждение образования

«Колледж бизнеса и права»

|  |
| --- |
| УТВЕРЖДАЮ  Заведующий  методическим кабинетом  \_\_\_\_\_\_\_\_\_\_\_\_\_Е.В.Фалей  « \_\_\_\_ » \_\_\_\_\_\_\_\_\_ 2016 |

|  |  |
| --- | --- |
| Специальность: 2-40 01 01 «Программное обеспечение информационных технологий» | Учебная практика по программированию |
| Составлена в соответствии с учебной программой, утвержденной директором Колледжа бизнеса и права 31.05.2016 | |

**Практическая работа №8**

**Инструкционно-технологическая карта**

Тема: Механизм регулярных выражений.

Цель: Сформировать умение работать с объектом System.Text.Regular Expressions.Regex.

Время выполнения: 2 часа.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Ознакомиться с теоретическими сведениями для выполнения работы.
2. Набрать, отладить, протестировать и выполнить примеры, предложенные в данной ИТК.
3. Выполнить индивидуальное задание согласно варианта, выданного преподавателем.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Механизм регулярных выражений в .NET Framework представлен классом [Regex](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex(v=vs.110).aspx). Механизм отвечает за синтаксический анализ и компиляцию регулярного выражения, а также за выполнение операций, которые сопоставляют шаблон регулярного выражения с входной строкой. Данный механизм — центральный компонент объектной модели регулярных выражений .NET Framework.

Можно использовать механизм регулярных выражений одним из двух способов:

* вызывая статичные методы класса [Regex](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex(v=vs.110).aspx). Параметры метода включают в себя входную строку и шаблон регулярного выражения. Механизм регулярных выражений кэширует регулярные выражения, которые используются в вызовах статичных методов, чтобы повторные вызовы статичных методов регулярных выражений, применяющие одно и то же регулярное выражение, выполнялись относительно быстро;
* создавая экземпляр объекта [Regex](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex(v=vs.110).aspx), передавая регулярное выражение конструктору класса. В этом случае объект [Regex](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex(v=vs.110).aspx) не изменяется (доступен только для чтения) и представляет механизм регулярных выражений, который тесно связан с одним регулярным выражением. Так как регулярные выражения, используемые экземплярами [Regex](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex(v=vs.110).aspx), не кэшируются, не следует создавать экземпляр объекта [Regex](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex(v=vs.110).aspx) несколько раз с одним и тем же регулярным выражением.

Можно вызывать методы класса [Regex](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex(v=vs.110).aspx), чтобы выполнить следующие операции:

* определить, соответствует ли строка шаблону регулярного выражения;
* извлечь одно сопоставление или первое сопоставление;
* извлечь все сопоставления;
* заменить сопоставленную подстроку;
* разделить одну строку на массив строк.

Эти операции описаны в далее.

**Пример 1. Сопоставление шаблона регулярного выражения**

Метод [Regex.IsMatch](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex.ismatch(v=vs.110).aspx) возвращает значение **true**, если строка соответствует шаблону; в противном случае возвращается значение **false**. Метод [IsMatch](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex.ismatch(v=vs.110).aspx) часто используется для проверки входной строки. Например, следующий код проверяет, содержит ли строка допустимый номер социального страхования для США.

using System;

using System.Text.RegularExpressions;

public class Example

{

public static void Main()

{

string[] values = { "111-22-3333", "111-2-3333"};

string pattern = @"^\d{3}-\d{2}-\d{4}$";

foreach (string value in values) {

if (Regex.IsMatch(value, pattern))

Console.WriteLine("{0} is a valid SSN.", value);

else

Console.WriteLine("{0}: Invalid", value);

}

}

}

**Пример 2. Извлечение одного сопоставления или первого сопоставления**

Метод [Regex.Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex.match(v=vs.110).aspx) возвращает объект [Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match(v=vs.110).aspx), который содержит сведения о первой подстроке, соответствующей шаблону регулярного выражения. Если свойство **Match.Success** возвращает значение **true**(найдено соответствие), можно получить информацию о последующих соответствиях, вызвав метод [Match.NextMatch](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match.nextmatch(v=vs.110).aspx). Этот метод можно вызывать, пока свойство **Match.Success** не вернет значение **false**. Например, следующий код использует метод [Regex.Match(String, String)](https://msdn.microsoft.com/ru-ru/library/0z2heewz(v=vs.110).aspx) для поиска первого экземпляра повторяющегося слова в строке. Затем вызывается метод [Match.NextMatch](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match.nextmatch(v=vs.110).aspx) для поиска других экземпляров.Пример просматривает свойство **Match.Success** после каждого вызова метода, чтобы определить, было ли текущее сопоставление успешным и следует ли вызывать метод [Match.NextMatch](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match.nextmatch(v=vs.110).aspx).

using System;

using System.Text.RegularExpressions;

public class Example

{

public static void Main()

{

string input = "This is a a farm that that raises dairy cattle.";

string pattern = @"\b(\w+)\W+(\1)\b";

Match match = Regex.Match(input, pattern);

while (match.Success)

{

Console.WriteLine("Duplicate '{0}' found at position {1}.",

match.Groups[1].Value, match.Groups[2].Index);

match = match.NextMatch();

}

}

}

**Пример 3. Извлечение всех сопоставлений**

Метод [Regex.Matches](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex.matches(v=vs.110).aspx) возвращает объект [MatchCollection](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.matchcollection(v=vs.110).aspx), содержащий сведения о всех сопоставлениях, найденных механизмом регулярных выражений во входной строке. Например, предыдущий пример можно изменить, чтобы вызывать метод [Matches](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex.matches(v=vs.110).aspx) вместо методов [Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex.match(v=vs.110).aspx) и [NextMatch](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match.nextmatch(v=vs.110).aspx).

using System;

using System.Text.RegularExpressions;

public class Example

{

public static void Main()

{

string input = "This is a a farm that that raises dairy cattle.";

string pattern = @"\b(\w+)\W+(\1)\b";

foreach (Match match in Regex.Matches(input, pattern))

Console.WriteLine("Duplicate '{0}' found at position {1}.",

match.Groups[1].Value, match.Groups[2].Index);

}

}

**Пример 4. Замена сопоставленной подстроки**

Метод [Regex.Replace](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex.replace(v=vs.110).aspx) заменяет каждую подстроку, соответствующую шаблону регулярного выражения, на указанную строку или шаблон регулярного выражения и возвращает всю замененную входную строку. Например, приведенный ниже код добавляет символ доллара США перед десятичным числом в строке.

using System;

using System.Text.RegularExpressions;

public class Example

{

public static void Main()

{

string pattern = @"\b\d+\.\d{2}\b";

string replacement = "$$$&";

string input = "Total Cost: 103.64";

Console.WriteLine(Regex.Replace(input, pattern, replacement));

}

}

**Пример 5. Разделение одной строки на массив строк**

Метод [Regex.Split](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex.split(v=vs.110).aspx) разделяет входную строку в позициях, заданных соответствием регулярного выражения. Например, следующий код помещает элементы нумерованного списка в массив строк.

using System;

using System.Text.RegularExpressions;

public class Example

{

public static void Main()

{

string input = "1. Eggs 2. Bread 3. Milk 4. Coffee 5. Tea";

string pattern = @"\b\d{1,2}\.\s";

foreach (string item in Regex.Split(input, pattern))

{

if (! String.IsNullOrEmpty(item))

Console.WriteLine(item);

}

}

}

// The example displays the following output:

// Eggs

// Bread

// Milk

// Coffee

// Tea

[**Объекты MatchCollection и Match**](javascript:void(0))

Методы Regex возвращают два объекта, входящие в объектную модель регулярного выражения: [MatchCollection](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.matchcollection(v=vs.110).aspx) и [Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match(v=vs.110).aspx).

Метод [Regex.Matches](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex.matches(v=vs.110).aspx) возвращает объект [MatchCollection](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.matchcollection(v=vs.110).aspx), содержащий объекты [Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match(v=vs.110).aspx), которые представляют все сопоставления, найденных механизмом регулярных выражений, в том порядке, в котором они присутствуют во входной строке. Если соответствий нет, метод возвращает объект [MatchCollection](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.matchcollection(v=vs.110).aspx) без членов. Свойство [MatchCollection.Item](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.matchcollection.item(v=vs.110).aspx) позволяет получить доступ к отдельным членам коллекции по индексу (от нуля до значения свойства [MatchCollection.Count](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.matchcollection.count(v=vs.110).aspx) минус 1). [Item](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.matchcollection.item(v=vs.110).aspx) — это индексатор коллекции (для C#). По умолчанию вызов метода [Regex.Matches](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex.matches(v=vs.110).aspx) использует отложенные вычисления для заполнения объекта [MatchCollection](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.matchcollection(v=vs.110).aspx). Доступ к свойствам, которым требуется полностью заполненная коллекция, например, свойства [MatchCollection.Count](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.matchcollection.count(v=vs.110).aspx) и [MatchCollection.Item](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.matchcollection.item(v=vs.110).aspx), может быть связан с дополнительными затратами. Поэтому рекомендуется обращаться к коллекции, используя объект [IEnumerator](https://msdn.microsoft.com/ru-ru/library/system.collections.ienumerator(v=vs.110).aspx), возвращаемый методом [MatchCollection.GetEnumerator](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.matchcollection.getenumerator(v=vs.110).aspx) (Оператор foreach в C#, который создает оболочку для интерфейса [IEnumerator](https://msdn.microsoft.com/ru-ru/library/system.collections.ienumerator(v=vs.110).aspx) коллекции).

**Пример 6. Использование метода Regex.Matches**

Метод [Regex.Matches(String)](https://msdn.microsoft.com/ru-ru/library/e7sf90t3(v=vs.110).aspx) используется для заполнения объекта [MatchCollection](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.matchcollection(v=vs.110).aspx) всеми соответствиями, найденными во входной строке. Пример перечисляет коллекцию, копирует соответствия в массив строк и записывает позиции символов в целочисленный массив.

using System;

using System.Collections.Generic;

using System.Text.RegularExpressions;

public class Example

{

public static void Main()

{

MatchCollection matches;

List<string> results = new List<string>();

List<int> matchposition = new List<int>();

// Create a new Regex object and define the regular expression.

Regex r = new Regex("abc");

// Use the Matches method to find all matches in the input string.

matches = r.Matches("123abc4abcd");

// Enumerate the collection to retrieve all matches and positions.

foreach (Match match in matches)

{

// Add the match string to the string array.

results.Add(match.Value);

// Record the character position where the match was found.

matchposition.Add(match.Index);

}

// List the results.

for (int ctr = 0; ctr < results.Count; ctr++)

Console.WriteLine("'{0}' found at position {1}.",

results[ctr], matchposition[ctr]);

}

}

// The example displays the following output:

// 'abc' found at position 3.

// 'abc' found at position 7.

Класс [Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match(v=vs.110).aspx) представляет результат одного сопоставления регулярного выражения. Доступ к объектам [Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match(v=vs.110).aspx) можно получить двумя способами:

* извлекая их из объекта [MatchCollection](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.matchcollection(v=vs.110).aspx), который возвращается методом [Regex.Matches](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex.matches(v=vs.110).aspx). Для извлечения отдельных объектов [Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match(v=vs.110).aspx) выполните итерацию коллекции, воспользовавшись конструкцией **foreach** либо используйте свойство [MatchCollection.Item](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.matchcollection.item(v=vs.110).aspx) для извлечения конкретного объекта [Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match(v=vs.110).aspx) по индексу или по имени. Также можно получить отдельные объекты[Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match(v=vs.110).aspx) из коллекции, циклически проходя по коллекции по индексу (от нуля до числа объектов в коллекции минус 1). Однако этот метод не использует отложенные вычисления, так как он обращается к свойству [MatchCollection.Count](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.matchcollection.count(v=vs.110).aspx).

Следующий пример извлекает отдельные объекты [Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match(v=vs.110).aspx) из объекта [MatchCollection](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.matchcollection(v=vs.110).aspx), циклически перебирая коллекцию с помощью конструкции **foreach** . Регулярное выражение просто сопоставляет строку "abc" во входной строке.

using System;

using System.Text.RegularExpressions;

public class Example

{

public static void Main()

{

string pattern = "abc";

string input = "abc123abc456abc789";

foreach (Match match in Regex.Matches(input, pattern))

Console.WriteLine("{0} found at position {1}.",

match.Value, match.Index);

}

}

// The example displays the following output:

// abc found at position 0.

// abc found at position 6.

// abc found at position 12.

* вызывая метод [Regex.Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex.match(v=vs.110).aspx), который возвращает объект [Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match(v=vs.110).aspx), представляющий первое соответствие в строке или ее части. Можно определить, найдено ли соответствие, получив значение свойства **Match.Success**. Чтобы извлечь объекты [Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match(v=vs.110).aspx), представляющие последующие соответствия, вызывайте метод [Match.NextMatch](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match.nextmatch(v=vs.110).aspx), пока свойство **Success** полученного объекта [Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match(v=vs.110).aspx) не будет иметь значение **false**.

Следующий пример использует методы [Regex.Match(String, String)](https://msdn.microsoft.com/ru-ru/library/0z2heewz(v=vs.110).aspx) и [Match.NextMatch](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match.nextmatch(v=vs.110).aspx) для сопоставления строки "abc" во входной строке.

using System;

using System.Text.RegularExpressions;

public class Example

{

public static void Main()

{

string pattern = "abc";

string input = "abc123abc456abc789";

Match match = Regex.Match(input, pattern);

while (match.Success)

{

Console.WriteLine("{0} found at position {1}.",

match.Value, match.Index);

match = match.NextMatch();

}

}

}

// The example displays the following output:

// abc found at position 0.

// abc found at position 6.

// abc found at position 12.

Два свойства класса [Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match(v=vs.110).aspx) возвращают объекты коллекции:

* свойство [Match.Groups](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match.groups(v=vs.110).aspx) возвращает объект [GroupCollection](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.groupcollection(v=vs.110).aspx), который содержит сведения о подстроках, соответствующих захватываемым группам в шаблоне регулярного выражения;
* свойство **Match.Captures** возвращает объект [CaptureCollection](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.capturecollection(v=vs.110).aspx) с ограниченным применением. Коллекция не заполняется для объекта [Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match(v=vs.110).aspx), свойство **Success** которого имеет значение **false**. В противном случае она содержит один объект [Capture](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.capture(v=vs.110).aspx), содержащий такие же данные, что и объект[Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match(v=vs.110).aspx).

Два дополнительных свойства класса [Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match(v=vs.110).aspx) предоставляют сведения о соответствии. Свойство **Match.Value** возвращает подстроку из входной строки, соответствующую шаблону регулярного выражения. Свойство **Match.Index** возвращает начальную позицию сопоставленной подстроки во входной строке с основанием ноль.

Класс [Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match(v=vs.110).aspx) также содержит два метода сопоставления шаблона:

* метод [Match.NextMatch](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match.nextmatch(v=vs.110).aspx) ищет совпадение после соответствия, представленного текущим объектом[Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match(v=vs.110).aspx), и возвращает объект [Match](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match(v=vs.110).aspx), представляющий соответствие;
* метод [Match.Result](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match.result(v=vs.110).aspx) выполняет указанную замену в сопоставленной строке и возвращает результат.

В следующем примере метод [Match.Result](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match.result(v=vs.110).aspx) используется для добавления символа $ и пробела перед каждым числом из двух дробных разрядов.

using System;

using System.Text.RegularExpressions;

public class Example

{

public static void Main()

{

string pattern = @"\b\d+(,\d{3})\*\.\d{2}\b";

string input = "16.32\n194.03\n1,903,672.08";

foreach (Match match in Regex.Matches(input, pattern))

Console.WriteLine(match.Result("$$ $&"));

}

}

// The example displays the following output:

// $ 16.32

// $ 194.03

// $ 1,903,672.08

Шаблон замены $$ $& указывает, что сопоставленную подстроку следует заменить на символ доллара ($) (шаблон $$), пробел и значение соответствия (шаблон $&).

[**Группы**](javascript:void(0))

Свойство [Match.Groups](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match.groups(v=vs.110).aspx) возвращает объект [GroupCollection](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.groupcollection(v=vs.110).aspx), содержащий объекты [Group](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.group(v=vs.110).aspx), которые представляют захватываемые группы в одном соответствии. Первый объект [Group](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.group(v=vs.110).aspx) в коллекции (объект с нулевым индексом) представляет все сопоставление. Каждый последующий объект представляет результаты одной захватываемой группы.

Можно извлечь отдельные объекты [Group](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.group(v=vs.110).aspx) из коллекции, используя свойство [GroupCollection.Item](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.groupcollection.item(v=vs.110).aspx). Неименованные группы можно извлечь по их номеру в коллекции, а именованные группы — по имени или номеру. Неименованные выделения идут в коллекции первыми и индексируются слева направо в том порядке, в котором они указаны в шаблоне регулярного выражения. Именованные выделения индексируются после неименованных слева направо в том порядке, в котором они указаны в шаблоне регулярного выражения. Чтобы определить, какие нумерованные группы доступны в коллекции, возвращаемой для определенного метода сопоставления регулярного выражения, вызовите метод [Regex.GetGroupNumbers](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex.getgroupnumbers(v=vs.110).aspx) экземпляра. Чтобы определить, какие именованные группы доступны в коллекции, вызовите метод [Regex.GetGroupNames](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.regex.getgroupnames(v=vs.110).aspx) экземпляра. Оба метода полезны в процедурах общего назначения, которые анализируют соответствия, найденные любым регулярным выражением.

Свойство [GroupCollection.Item](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.groupcollection.item(v=vs.110).aspx) — это индексатор коллекции для C#. Это значит, что доступ к отдельным объектам [Group](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.group(v=vs.110).aspx) можно получить по индексу (или, для именованных групп, по имени) следующим образом:

Group group = match.Groups[ctr];

**Пример 7. Использование конструкций группировки**

В следующем примере определяется регулярное выражение, которое использует конструкции группировки для выделения месяца, дня и года из даты.

using System;

using System.Text.RegularExpressions;

public class Example

{

public static void Main()

{

string pattern = @"\b(\w+)\s(\d{1,2}),\s(\d{4})\b";

string input = "Born: July 28, 1989";

Match match = Regex.Match(input, pattern);

if (match.Success)

for (int ctr = 0; ctr < match.Groups.Count; ctr++)

Console.WriteLine("Group {0}: {1}", ctr, match.Groups[ctr].Value);

}

}

// The example displays the following output:

// Group 0: July 28, 1989

// Group 1: July

// Group 2: 28

// Group 3: 1989

**Пример 8. Использование захватываемой группы**

Класс [Group](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.group(v=vs.110).aspx) представляет результат одной захватываемой группы. Объекты Group, представляющие захватываемые группы, заданные в регулярном выражении, возвращаются свойством [Item](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.groupcollection.item(v=vs.110).aspx) объекта[GroupCollection](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.groupcollection(v=vs.110).aspx), который возвращается свойством [Match.Groups](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.match.groups(v=vs.110).aspx). Свойство [Item](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.groupcollection.item(v=vs.110).aspx) — это индексатор (для C#) класса [Group](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.group(v=vs.110).aspx).

В следующем примере вложенные конструкции группировки используются для записи подстрок в группы. Шаблон регулярного выражения (a(b))c сопоставляет строку "abc". Он назначает подстроку "ab" первой захватываемой группе, а подстроку "b" — второй захватываемой группе.

List<int> matchposition = new List<int>();

List<string> results = new List<string>();

// Define substrings abc, ab, b.

Regex r = new Regex("(a(b))c");

Match m = r.Match("abdabc");

for (int i = 0; m.Groups[i].Value != ""; i++)

{

// Add groups to string array.

results.Add(m.Groups[i].Value);

// Record character position.

matchposition.Add(m.Groups[i].Index);

}

// Display the capture groups.

for (int ctr = 0; ctr < results.Count; ctr++)

Console.WriteLine("{0} at position {1}",

results[ctr], matchposition[ctr]);

// The example displays the following output:

// abc at position 3

// ab at position 3

// b at position 4

В следующем примере именованные конструкции группировки используются для выделения подстрок из строки, содержащей данные в формате "DATANAME:VALUE", которые регулярное выражение разделяет в позиции двоеточия (:).

Regex r = new Regex("^(?<name>\\w+):(?<value>\\w+)");

Match m = r.Match("Section1:119900");

Console.WriteLine(m.Groups["name"].Value);

Console.WriteLine(m.Groups["value"].Value);

// The example displays the following output:

// Section1

// 119900

Свойства класса [Group](https://msdn.microsoft.com/ru-ru/library/system.text.regularexpressions.group(v=vs.110).aspx) предоставляют сведения о захватываемой группе: свойство Group.Value содержит выделенную подстроку, свойство Group.Index указывает начальную позицию захватываемой группы во входном тексте, свойство Group.Length содержит длину выделенного текста, а свойство Group.Success указывает, соответствует ли подстрока шаблону, заданному захватываемой группой.

ВАРИАНТЫ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ

**Общая постановка задачи**. Выполнить задание, применив в решении регулярные выражения.

1. Найти количество строк текстового файла, которые заканчиваются словом, являющимся правильным MAC-адресом. MAC-адрес состоит из шести двузначных 16-ричных чисел, разделённых двоеточием. Регистр букв не различается. Пример правильного MAC-адреса: aE:dC:cA:56:76:54
2. В текстовом файле некоторые слова являются IP-адресами компьютеров в формате d.d.d.d, где d - целое число из диапазона от 0 до 255. Вывести все такие слова (по одному в строку) в новый файл.
3. Некоторые строки текстового файла содержат GUID (строчка, состоящая из 8, 4, 4, 4, 12 шестнадцатеричных цифр, разделенных символом «минус»). Регистр букв не различается. Создать на основе входного файла новый текстовый файл, не включая в него найденные строки. Пример GUID: e02fd0e4-00fd-090A-ca30-0d00a0038ba0
4. В текстовом файле могут содержаться (как отдельные слова) IP-адреса компьютеров в формате d.d.d.d, где d - целое число из диапазона от 0 до 255. Создать новый файл, удаляя IP-адреса, в которых последнее число d начинается с заданной цифры (данная цифра вводится с клавиатуры).
5. Найти количество строк текстового файла, которые начинаются словом, являющимся правильной датой в формате dd/mm/yyyy. Диапазон лет - от 1800 до 2100. Правильно обрабатывать високосные года.
6. В текстовом файле могут содержаться (как отдельные слова) MAC-адреса (см. вариант 1). Вывести все такие слова, третье число в которых равно искомому, в новый файл (по одному в строку). Искомое число вводится с клавиатуры в десятичной системе счисления.
7. В текстовом файле могут содержаться (как отдельные слова) числа в римской системе счисления. Вывести значение максимального из таких слов в десятичной системе счисления. Если таких слов нет, вывести ноль.
8. Найти количество строк текстового файла, которые содержат не менее двух слов, являющихся правильной записью времени в формате HH:MM:SS
9. В текстовом файле могут содержаться (как отдельные слова) числа в римской системе счисления. Вывести значения всех таких слов в десятичной системе счисления в порядке возрастания. Если таких слов нет, вывести соответствующее сообщение.
10. Найти количество строк текстового файла, которые не содержат слов, являющихся правильной записью времени в 12-часовом формате (часы и минуты разделяются двоеточием, секунды отсутствуют).
11. «Хороший» пароль должен иметь длину в 8 символов, содержать большие, маленькие буквы латинского алфавита, цифры, подчёркивание, причём должен быть включён хотя бы один символ из каждой группы. Найти количество «хороших» паролей в текстовом файле.
12. Формат даты dd MMM yy содержит: номер дня без ведущего нуля, пробел, трёхсимвольное английское обозначение месяца (прописными буквами), пробел, две последние цифры года. В текстовом файле могут содержаться (как отдельные слова) записи дат в таком формате. Найти максимальную дату, предполагая, что все они относятся к XX столетию. Если таких слов нет, вывести соответствующее сообщение.
13. Автомобильный номер Германии состоит из кода региона (от одной до трёх прописных латинских букв), пробела, кода серии (одна или две прописных латинских буквы) и собственно номера (четырёхзначного числа без ведущих нулей). Серии SS и SD не используются, как напоминающие о нацистском прошлом. Текстовый файл содержит список автомобильных номеров, по одному в строке. Найти количество ошибочных номеров.

ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Албахари, Дж. C#5.0. Справочник. Полное описание языка / Дж. Албахари, Б. Албахари. – 5-е изд. – М.: ООО «И.Д. Вильямс», 2014. – 1008 с.: ил.
2. Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5 / Э. Троелсен. – 6-е изд. – М.: ООО «И.Д. Вильямс», 2013. – 1312 с.: ил.
3. Шилдт, Г. C#4.0: Полное руководство / Г. Шилдт. – М.: ООО «И.Д. Вильямс», 2011. – 1056 с.: ил.
4. Гойвертс, Я. Регулярные выражения. Сборник рецептов / Я. Гойвертс, Ст. Левитан. – 2-е изд. – СПб.: Символ-Плюс, 2015. – 704 с.: ил.
5. Фицджеральд, М. Регулярные выражения: основы / М. Фицджеральд. – М.: ООО «И.Д. Вильямс», 2015. – 144 с.: ил.
6. Полное руководство по языку программирования С# 6.0 и платформе .NET 4.6 [Электронный ресурс] / Сайт о программировании. – metanit.com, 2012-2016. – Режим доступа: [http://metanit.com/sharp/ tutorial](http://metanit.com/sharp/%20tutorial). – Дата доступа: 20.08.2016.

Преподаватель Е.В. Багласова

|  |
| --- |
| Рассмотрено на заседании цикловой  комиссии программного обеспечения информационных технологий №10  Протокол №\_\_\_\_\_от «\_\_\_»\_\_\_\_2016  Председатель ЦК *( )* Т.Г.Багласова |