Поиск и замена текста при помощи регулярных выражений

System.Text.RegularExpressions — пространство имен для работы с регулярными выражениями в C#

Основные классы для работы с регулярными выраденииями:

- **Regex** – постоянное регулярное выражение. В конструктор при создании передается шаблон
- **Match** – результат очередного применения регулярного выражения к строке
- **MatchCollection** – коллекция всех совпадений
- **Group** – результат для одной группы
- **GroupCollection** – коллекция найденных групп в одном сопоставлении

Алгоритм работы с регулярными выражениями:

1) создать входную строку и строку-шаблон

```
string input = "… ";       // входная строка
string pattern = @"… "; // шаблон. @ в начале для упрощения записи (чтобы не экранировать символы)
```

2) создать объект класса Regex

```
Regex regex = new Regex(pattern);
```

В конструкторе после шаблона можно указать опции RegexOption, напрмимер:

```
regex = new Regex(pattern, RegexOption.IgnoreCase); // игнорировать регистр
```

3) применить регулярное выражение для поиска или замены совпадений

Пример 1. Проверка, что строка соответствует шаблону

```
if (regex.IsMatch(input)) // IsMatch — возвращает истину, если совпадение найдено
{
        // в строке есть соответствие шаблону
}
```

Пример 2. Получение всей подстроки (совпадения), соответствующей шаблону

```
Match match = regex.Match(input);
if (match.Success)
{
        // работа с найденной подстрокой (хранится в match.Value)
}
```

Пример 3. Перебор всех групп найденной подстроки

```
Match match = regex.Match(input);
if (match.Success)
{
        // работа с match.Groups (коллекция всех групп)
        // работа с определенной группой (хранится в match.Groups[номер или имя группы].Value)
        // match.Groups[0].Value — вся найденная подстрока
}
```

Пример 4. Перебор всех совпадений, соответствующих шаблону

```
Match match = regex.Match(input);
while (match.Success) // перебор всех совпадений, пока они есть
{
        // работа с найденной подстрокой (хранится в match.Value)
        // работа с найденной группой (хранится в match.Groups[номер или имя].Value)

        match = match.NextMatch(); // переход к следующему совпадению
}
```

Пример 5. Перебор всех совпадений, соответствующих шаблону

1 вариант

```
MatchCollection matches = regex.Matches(input);
for (int i = 0; i < matches.Count; i++)
{
        //…
}
```

2 вариант

```
foreach (var match in regex.Matches(input))
{
        //…
}
```

Замена подстрок

Метод Replace позволяет заменить строку, соответствующую регулярному выражению, другой строкой.

1) создать входную строку, строку-шаблон и шаблон замены

```
string input = "… ";              // входная строка
string pattern = @"… ";           // шаблон
string replacement = "… ";        // шаблон замены
```

2) вызвать Replace у регулярного выражения

```
string result = Regex.Replace(input, pattern, replacement);
```

или

```
Regex regex = new Regex(pattern);
string result = regex.Replace(input, replacement);
```

Разделение на массив строк

Метод Split позволяет получить из строки массив строк. Разделитель — регулярное выражение.

1) создать входную строку и строку-шаблон

```
string input = "… ";       // входная строка
string pattern = @"… "; // шаблон
```

2) вызвать Split у регулярного выражения

```
string[] result = Regex.Split(input, pattern);
```