УДК 378.001.76

ИСПОЛЬЗОВАНИЕ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ ДЛЯ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ РАБОТЫ С ТЕКСТОМ

Э. И. ЯСЮКОВИЧ
Белорусско-Российский университет
Могилев, Беларусь

Регулярное выражение – это формальный язык поиска и манипуляций с подстроками текста, основанный на использовании шаблонов, представляющих собой строки, состоящие из символов и метасимволов. То есть это своеобразный фильтр для текстовых данных. Они используются в языках программирования, офисных и некоторых других программных средствах.

Изначально регулярные выражения появились в среде UNIX и использовались в языке программирования Perl. Затем они были внедрены в Windows, где поддерживаются множеством классов .NET из пространства имен System.Text.RegularExpressions.

В качестве используемых в регулярных выражениях метасимволов можно выделить их классы: […] – любой из указанных в скобках символ; [^…] – любой из неуказанных в скобках символ; **.** – любой символ, кроме перевода строки или другого разделителя Unicode-строки; \w – любой текстовый символ, не являющийся пробельным; \W – любой символ, не являющийся текстовым; \s – любой пробельный символ из набора Unicode; \S – любой непробельный символ из набора Unicode; \d – любая цифра; \D – любой символ, отличный от цифры, эквивалентно [^0-9]. В регулярных выражениях используются также группы метасимволов повторения, выбора и якорные символы.

Работа с регулярными выражениями требует некоторых предварительных настроек, описаний и технологий. Например, использование их при работе в текстом в редакторе MS Word начинается с вызова окна «Найти и заменить», которое открывается нажатием «Ctrl+F» или кнопки «Заменить» на вкладке «Главная». Затем в появившемся окне «Найти и заменить» необходимо нажать кнопку «Больше >>» и отметить опцию «Подстановочные знаки», чтобы Word воспринимал регулярные выражения, т. е., чтобы служебные символы в полях «Найти» и «Заменить» воспринимались как подстановочные знаки, а не как символы, которые необходимо найти.

После этого в поле ввода «Найти» необходимо ввести регулярное выражение, описывающее искомое слово или подстроку текста, а в поле «Заменить на» – выражение, описывающее выполняемые действия с найденными фрагментами текста. Например, чтобы заменить строку «Иванов М. С.» на «М. С. Иванов» достаточно построить регулярное выражение:

([А-Я]{1;1}[а-я]{1;55}) ([А-Я]{1;1})\. ([А-Я]{1;1})\. ,

которое необходимо ввести в поле ввода «Найти». Здесь [А-Я]{1;1} означает, что первым символом в искомой фамилии может быть одна русская прописная буква; [а-я]{1;55} – указывает, что фамилия может

содержать от 2 до 55 и более строчных букв; после которой – один пробел; ([А-Я]{1;1})\. – после пробела одна прописная буква с точкой, которая повторяется два раза через пробел.

В поле «Заменить на» введено выражение: \2.\3.^s\1 . Здесь «\2.» означает вторую подстроку в исходном тексте, т. е. «М.», «\3.» – третью подстроку, т. е. «С.», «^s\1» – указывает, что эти строки необходимо разместить в начале.

В языке C# также имеются средства для работы со строками текста, например, в пространстве имен System.Text имеется класс String, который предоставляет достаточную функциональность. Однако в этом языке для работы с текстами имеется мощный инструмент RegularExpression – регулярные выражения, представляющий собой основанный на использовании метасимволов формальный язык по обработке больших текстов, т. е. по выполнению манипуляций с его подстроками. Основная функциональность регулярных выражений в этом языке сосредоточена в пространстве имен System.Text.RegularExpressions, а центральным классом для работы с ними является класс Regex, в котором содержатся методы: IsMatch(), Match(), Matches(), Replace() и Split().

Класс Regex имеет ряд конструкторов, позволяющих выполнить начальную инициализацию объекта. Эти конструкторы в качестве одного из параметров принимают перечисление `RegexOptions`, которое может принимать значения CultureInvariant, ExplicitCapture, IgnoreCase, IgnorePatternWhitespace, Multiline, RightToLeft, Singleline.

Простейший пример кода C# программы, выполняющей работу с регулярными выражениями, может иметь вид:

```
using System.Text.RegularExpressions;
namespace RegexSample  {
Class mainClass                {
Static void Main()   {
String text = "Метод <b>Replace</b>используется для поиска с заменой";
Bool match = Regex.IsMatch(text, "<b>(.*?)</b>");          // Regex – класс
Console.WriteLine(match.ToString());
tring textOut = Regex.Replace(text, "<b>(.*?)</b>", @"<u>$1</u>",
RegexOptions.IgnoreCase);
Console.WriteLine(textOut);
string[] splitStr = Regex.Split(text, ">");
for (int i = 0; i < splitStr.Length; i++)            {
Console.WriteLine(splitStr[i]);                                }
MatchCollection matches = Regex.Matches(text, "<b>(.*?)</b>",
RegexOptions.IgnoreCase);
Console.WriteLine("Совпадений: " + matches.Count.ToString());
For (int i = 0; i < matches.Count; i++)  {
Console.WriteLine("Совпадение " + i + ": " + matches[i].Value);        }
Console.ReadLine();          }
}
```