5 Тұрақты өрнектер

5.1 Тұрақты өрнектер түсінігі

Бағдарламалаудың кез келген тілінде мәтіндік ақпаратпен жұмыс істеу үшін арнайы құралдар бар. Әдетте, олар string, char типіндегі айнымалылар және осы айнымалылармен жұмыс істеуге арналған операторлар, функциялар немесе кластардың жиынтығы.

Тұрақты өрнектердің мәтіндік ақпаратты өңдеуге арналған бағдарламалаудың мамандандырылған тілі болып келеді.

.NET кітапханасының тұрақты өрнектері Perl5 тілінің тұрақты өрнектеріне негізделген және де өзіне оның функционалдық мүмкіншіліктерінің үлкен бөлігін енгізеді.

Әрине, тұрақты өрнектермен есептелініп шешілетін кез келген есеп System.String және System.Text.StringBuilder кластары әдістерінің көмегімен жүзеге асырылуы мүмкін, бірақ осы шешімдердің кодтарының көлемдері тұрақты өрнектердің көмегімен жазылған код көлемінен бірнеше есе үлкен болады.

Жүйелік бағдарламалаушылармен тұрақты өрнектердің кейбір элементтерін операциялық жүйлермен «командалық жол» режімінде жұмыс істегенде жемісті қолданылады.

5.2 Тұрақты өрнектермен шешілетін негізгі міндеттер

.NET кітапханасының тұрақты өрнектері мәтіндік ақпаратпен жұмыс істеу кезінде келесі негізгі міндеттердің шешімін қамтамасыз етеді:

мәтінде берілген фрагменттің болуын тексеру;

мәтінді енгізуге қол жетімділігін тексеру, мысалы, парольдерді;

мәтінде берілген шаблон бойынша іздестіру;

мәтін фрагменттерін редакциялау, ауыстыру және жою;

кез келген құрылымды файлдарынан деректерді алу, мысалы, HTML-беттері;

мәтінмен жұмыс нәтижелері бойынша қорытынды есеп берулерді қалыптастыру.

Әрине, бұл тұрақты өрнектердің көмегімен шешілетін міндеттердің бір бөлігі ғана. Олардың кейбіреулерін біздің дәрістерімізде қарастыратын боламыз.

Сонда C# тіліндегі жолдар өзгерілмейтін объектілер болатынын естен шығармау керек, сонымен тұрақты өрнектер бастапқы жолды өзгерте алмайды.

5.3 Тұрақты өрнектер тілінің символикасы

Тұрақты өрнектер тілінің символикасы қарапайым және метасимволдар сияқты екі түрлі символдармен берілген.

Мәтінде қарапайым символдар өздерін өздері көрсетеді, мысалы, мәтінде іздестіру кезінде пайдаланылатын «Само*» шаблоны, осы шаблон «Сам»

символымен басталатын сөздерді іздейтін болады. Мысалы, «Сам», «Само» немесе «Самоо». «*» символы метасимвол болып табылады және алдыңғы «о» символы 0 немесе одан да көп рет қайталануы мүмкін екенін білдіреді. Сонымен, қарапайым символдар мәтіндік ақпаратты жазуға мүмкіндік береді.

Метасимволдарда символдық мән емес арнайы мән бар.

Әрине, тұрақты өрнектер тілінде «резервтелген» символдар бар, олардың пайдаланылуы тілмен символ сияқты емес метасимвол ретінде қабылданады. Мысалы, алдыңғы мысалдағы «*» символы.

Егерде мәтінде * метасимволын емес «*» символын пайдалану қажет болса, онда оны пайдалану алдында «\» символын орнату қажет, мысалы, З*а – З-ті а-ға көбейту.

Егерде әріпті метасимвол ретінде пайдалану қажет болса, онда оны пайдалану алдына «\» символын орнату қажет, мысалы, «Lab\d» жазбасы мәтінде «\d» орнына сияқты кез келген ондық цифр бола алатынын білдіреді - Lab1 немесе Lab6.

Метасимволдар өте көп. Әдетте, әдебиетте метасимволдар олардың функционалдық тағайындалуларына байланысты топтармен қарастырылады. Мысалы, анықтаушы метасимволдар, қайталағыштар немесе квантификаторлар, ауыстырғыштар немесе «Символдар кластары» және т.б. Олардың нақты зерделенуі үшін әдетте дәрістер емес анықтамалық әдебиет пайдаланылады. Бірақ, көптеген студенттер «анықтамалық әдебиетті оқи алмайды» және оқытушылар дәрістерінде дәстүрлі түрде тұрақты өрнектер тілінің негізгі метасимволдарын келтіруге мәжбүрлі. Дәстүрлерден шегінбестен, олардың функционалдық бағытталуына байланысты кейбір метасимволдарды қарастырамыз.

5.4 Қайталағыштар

Қайталағыш метасимволдары (әдебиетте оларды кейде квантификаторлар деп атайды) қарапайым символдан немесе символдар класынан кейін орналасады және мәтіндегі мүмкін болатын қайталаулардың санын анықтайды.

5.1 кестесінде келтірілген ең жиі пайдаланылатын қайталағыштардың қарастырамыз.

5.1-кесте
Қайталағыштар

Метасимвол	Сипаттамасы	Мысал
*	Алдыңғы элементтің нөл немесе одан да көп рет қайталануы	Выражение ca*t соответствует фрагментам ct, cat, caat, caaaaaaaaaat и т. д.
+	Алдыңғы элементтің бір немесе одан да көп рет қайталануы	Выражение ca+t соответствует фрагментам cat, caat, caaaaaaaaaat и т. д.
?	Алдыңғы элементтің нөл немесе бір рет	Выражение ca?t соответствует фрагментам ct и cat

	қайталануы	
{n}	Алдыңғы элементтің тура n рет қайталануы	Выражение ca{3}t соответствует фрагменту caaat, а выражение (cat){2} — фрагменту catcat
{n,}	Алдыңғы элементтің тым болмағанда n рет қайталануы	Выражение ca{3,}t соответствует фрагментам caaat, caaaat, caaaaaaaaaaat и т. д.
{n,m}	Алдыңғы элементтің n реттен m ретке дейін қайталануы	Выражение ca{2,4}t соответствует фрагментам caat, caaat и caaaat

Келтірілген мысалдардан «қайталағыштар» метасимволдары мәтіннің сәйкес орнында метасимвол алдында орналасқан символдың мүмкін болатын қайталануының санын анықтайтынын көреміз.

5.5 Анықтаушы метасимволдар

Осы топтың метасимволдары мәтін жолындағы тұрақты өрнекпен сәйкес келуді іздейтін орынды анықтайды. Жиі пайдаланылатын анықтаушы метасимволдар 5.2 кестесінде келтірілген.

5.2-кесте
Анықтаушы метасимволдар

Метасимвол	Сипаттама
^	Тұрақты өрнекпен сәйкес келетін фрагментті тек жол басынан іздеу керек
$	Тұрақты өрнекпен сәйкес келетін фрагментті тек жол соңынан іздеу керек
\A	Тұрақты өрнекпен сәйкес келетін фрагментті тек көп жолды жолдың басынан іздеу керек
\Z	Тұрақты өрнекпен сәйкес келетін фрагментті тек көп жолды жолдың соңынан іздеу керек
\b	Тұрақты өрнекпен сәйкес келетін фрагмент сөз шегінде басталады немесе аяқталады (яғни, \w және \W метасимволдарына сәйкес келетін сөздер арасында)
\B	Тұрақты өрнекпен сәйкес келетін фрагмент сөз шегінде кездеспеуі керек

Мысалы, ^cat өрнегі жол басында кездесетін cat символдарына, cat$ өрнегі — жол соңындағы cat символдарына сәйкес келеді (яғни оның соңында жолды аудару символы орналасқан), ал ^$ өрнегі бос жол, яғни бұл оның артында лезде жол соңы болатын жол басы болып табылады.

5.5 Ауыстырғыштар немесе «Символдар кластары»

Осы топтың метасимволдары мәтіндегі символдардың нақты тізбегі үшін ауыстырғыштар сияқты әрекет етеді. Жиі қолданылатын ауыстырғыштар 5.3 кестесінде келтірілген [б.356].

5.3-кесте.
Ауыстырғыштар немесе «Символдар кластары»

Символдар класы	Сипаттама	Мысал
.	Кез келген символ, \n басқа	Выражение c.t соответствует фрагментам cat, cut, c1t, c{t и т. д.
[]	Жақша ішінде жазылған тізбекке кіретін кез келген бірлік символ. Символдар диапазонын қолдануға рұқсат етілген	Выражение c[au1]t соответствует фрагментам cat, cut и c1t, а выражение c[a-z]t — фрагментам cat, cbt, cct, cdt, ..., czt
[^]	Жақша ішінде жазылған тізбекке кірмейтін кез келген бірлік символ. Символдар диапазонын қолдануға рұқсат етілген	Выражение c[^au1]t соответствует фрагментам cbt, c2t, cXt и т. д., а выражение c[^a-zA-Z]t — фрагментам cит, c1t, cЧt, c3t и т. д.
\w	Кез келген алфавиттік-цифрлік символ, яғни бас және кіші әріптер және ондық цифрлардың жиынтығынан қандайда бір символ	Выражение c\wt соответствует фрагментам cat, cut, c1t, cЮt и т. д., но не соответствует фрагментам c{t, c;t и т. д.
\W	Кез келген алфавиттік-цифрлік емес символ, яғни бас және кіші әріптер және ондық цифрлардың жиынтығына кірмейтін қандайда бір символ	Выражение c\Wt соответствует фрагментам c{t, c;t, c t и т. д., но не соответствует фрагментам cat, cut, c1t, cЮt и т. д.
\s	Кез келген пробелдік символ, мысалы, пробел символы, табуляция (\t, \v), жол ауыстыру (\n, \r), жаңа бет (\f)	Выражение \s\w\w\w\s соответствует любому слову из трех букв, окруженному пробельными символами
\S	Кез келген пробелдік емес символ, яғни пробелдік жиынтыққа кірмейтін символ	Выражение \s\S\S\s соответствует любым двум непробельным символам, окруженным пробельными
\d	Кез келген ондық сан	Выражение c\dt соответствует фрагментам c1t, c2t, …, c9t
\D	Кез келген ондық сан емес символ	Выражение c\Dt не

соответствует фрагментам
c1t, c2t, …, c9t

5. Арнайы (басқарушы) символдар

Осы символдарды біз бірінші семестрде C# тілінде бағдарламалау негіздерін зерделегенде қарстырғанбыз, бірақ әдістімелік тұрғыдан оларды осы дәрісте қайталған дұрыс. Негізінен осы символдар компьютердің сырқы құрылғысына ақпаратты шығаруды басқару үшін пайдаланылады, мысалы, мониторға. Ең көп пайдаланылатын арнайы символдар 5.4 кестесінде келтірілген.

5.4-кесте.
Тұрақты өрнектердің кейбір арнайы символдары

\a	Дыбыстық сигнал (ескерту)
\b	Бір позицияға қайтару символы
\t	Көлденең табуляция
\r	Каретканы қайтару (каретка- жылжып тұратын бөлік)
\v	Тік табуляция
\f	Бетті ауыстыру
\n	Жаңа жол символы (жолды ауыстыру)
\e	Escape-символы

Ерекше жағдай: тұрақты өрнегі ішінде \b тізбегі [ ] шаршы жақшаларына алынған жиынтықтан басқа сөз шегін білдіреді, ол *забой* символын білдірерді.

5.7 Тұрақты өрнектер

Тұрақты өрнектер тілінің көмегімен жазылған символдар тізбегін тұрақты өрнектер дейміз. Әдетте, бұл қандай да бір мәтінде фрагменттерді іздестіру үшін пайдаланылатын қандай да бір үлгі. Мысалы, келесі қарапайым тұрақты өрнек бүтін сандарды сипаттау үшін арналған:

[–+]?\d+

Мұнда, «[–+]» нұсқауы осы жерде «жақшалар ішінде тізбекке жазылған кез келген бірлік символ» орналаса алатынын білдіреді, ал «?» нұсқауы – осы символ «бір рет емес немесе бір рет ғана» кездесе алатынын білдіреді. «\d» нұсқауы осы жерде «кез келген ондық цифр» болуы мүмкін екендігін білдірсе, ал «+» нұсқауы, «алдыңғы элементтің бір немесе одан да көп қайталануы», яғни цифрдің қайталануы мүмкін екендігін білдіреді.

Нақты сандарды сипаттау үшін арналған тұрақты өрнек келесі түрге ие:

[–+]?\d+\.?\d*

Мұнда нақты санда «.» символы бар болуы мүмкін екендігі және бөлшекті бөліктің цифры «алдыңғы элементі нөл немесе одан да көп рет қайталануы» бар болуы мүмкін екендігі көрсетілген нұсқауы қосылған.

Тұрақты өрнектермен жұмыс істеуге арналған барлық типтер System.Text.RegularExpressions атаулар кеңестігінде орналасқан.

Regex класы осы атаулар кеңестігінің басты класы, оның данасы тұрақты өрнекті көрсетеді. Regex класы String класы сияқты өзгермейтін болып табылады, яғни оның данасын құрған соң түзету рұқсат етілмейді.

Regex класының негізгі қасиеттері Index және Length, оларда айқындалған сәйкес келудің орнының бастапқы мәні және ұзындығы бар болады. Табылған мән Value қасиетіне орналастырылады.

IsMatch, Match және Matches әдістері мәтінде іздестіруді атқаратын негізгі әдістер.

Regex класының IsMatch әдісі мәтінде іздестіру жүргізеді және берілген мәтінде тұрақты өрнекке сәйкес келетін фрагмент табылса, true қайтарады, ал сәйкес келетін фрагмент табылмаса false қайтарады.

Regex класының Match әдісі, IsMatc әдісіне қарағанда, берілген тұрақты өрнекпен сәйкес келген мәтіннің бірінші фрагментінің Match класының объектісін қосымша құрады.

Regex класының Matches әдісі берілген тұрақты өрнекпен сәйкес келген мәтіннің барлық фрагменттерінің топтамасы (массив) MatchCollection / класының объектісін қайтарады.

Қандай да бір мәтінде іздестірудегі осы әдістердің жұмысын қарастырайық:

```
using System;
using System.Text.RegularExpressions;
using System.Text;

namespace ConsoleApplication1
{
 class Program
 {
  static void Main()
  {
   string text    = "Язык регулярных выражение определяет шаблоны
символов.";
   if (Regex.IsMatch(text, ”ред|выр")) Console.WriteLine("Есть
совпадение!");    else Console.WriteLine("Совпадений нет.");
   Console.WriteLine();
   Match Obj = Regex.Match(text, @”ред|выр");
   Console.WriteLine( Obj.Success);
   Console.WriteLine( Obj.Index);
   Console.WriteLine( Obj.Length);
   Console.WriteLine( Obj.Value);
   Console.WriteLine( Obj.ToString());
   //Console.WriteLine(Obj);
   Console.WriteLine();
   MatchCollection Kol = reg.Matches(text);
   Console.WriteLine("Число совпадений = {0}", Kol.Count);
   foreach(Match T in Regex.Matches(text, "ред|выр"))
    Console.WriteLine("{0} - {1}",T.Index, T.Value);
   Console.ReadLine();
  }
```

```
  }
}
```
Работа программы:
Есть совпадение!

True
16
3
выр
выр

Число совпадений = 2
16 – выр
28 – ред

Бағдарламаға деген кейбір түсініктемелер.

Regex класының IsMatch әдісін пайдаланғанда, біз қолдануға тек іздестірудің логикалық нәтижесін true немесе false аламыз.

Regex класының Match әдісін пайдалану кезінде біз өрістерінің мәнін монитор экранына шығаратын объект құрастырылады. Жолдық түрге түрлендірілген бүкіл объектінің мәні жеке жол ретінде шығарылған. Осы әдіс үшін соңғы шығару жолы өзгертілді:

```
Console.WriteLine(Obj);
```

Қателер жөнінде хабарламалар жоқ және еш өзгеріс жоқ.

Осы әдісте тұрақты өрнек алдында «@» символы орнатылған. Бұл «\» бэкслеш символынан басталатын басқарушы символдарда пайдалану кезінде C# тілінде әрекет ететін «экрандау» механизмін айналып өту үшін істелінген. Біздің тұрақты өрнекте «\» символдары жоқ, сондықтанда «@» символын қоймауға болады. Тұрақты өрнектің мәтінінде «@»префиксінсіз және «\» символының болуынсыз әр «\» символын кері қисық сызық деген төрт символмен «экрандау» қажет болушы еді. Мысалы, бізге бэкслешы бар тұрақты өрнекті - \sector жазу қажет. Бірақ, «\s» комбинациясы ауыстырғыш болып табылады, сондықтанда «\» символын экрандау қажет және де жолда «\\sector» қосарланған символ пайда болады. Бірақ, тұрақты өрнекте екі бэкслеште қайтадан экрандалуға тиіс, яғни "\\\\sector". Бір сөзбен айтқанда, бэкслешті салыстыру үшін тұрақты өрнек жолы ретінде '\\\\' жазу керек, өйткені тұрақты өрнек \\ болуға тиіс, және әр қисық сызық \\ сияқты қарапайым жолға ауыстырылуға тиіс.

Тұрақты өрнектермен жұмыс істегенде, «@» символын қолдану ұсынылады – яғни тұрақты өрнектерді @-константалар түрінде көрсету.

Regex класының Matches әдісін пайдаланғанда, қасиеттерін foreach циклының көмегімен қарастыруға болатын Match типті объектілердің массиві құрастырылады.

Біздің мысалда тұрақты өрнектер Regex класының әдістерінде талданатын мәтіннен кейін екінші параметр ретінде берілген болатын. Бірақ, Regex класының объектілерін құру (тұрақты өрнекті құру) және осы объект үшін

аталған әдістерді пайдалану нұсқасы бар. Бұл жағдайда әдістерде тек бір формалды параметр бар – талданатын мәтін. Мысалы:

```
Regex reg = new Regex(@"ред|выр");
Match Obj = reg.Match(text);
```

Regex класында тұрақты өрнекті сипаттау үшін бірнеше артық жүктелген конструктор анықталған:

Regex() – бос тұрақты өрнекті құрайды;

Regex(String) – берілген тұрақты өрнекті құрайды;

Regex(String, RegexOptions) – берілген тұрақты өрнекті құрайды және RegexOptions (опциялар) тізбелеудің элементтерінің көмегімен оны өңдеу үшін параметрлер береді.

Regex класында мәтінде іздестіру әдістерімен қатар, мәтінді өңдеудің басқа да көптеген әдістері бар екенін атап кету қажет. Мысалы, Regex.Replace әдісі берілген тұрақты өрнекке сәйкес мәтін фрагменттерін ауыстырады, ал Regex.Split әдісі тұрақты өрнектерге мәтінді бөлушіні береді.

Regex класының әр әдісі анықтаушы опцияларды қолдана алады, олардың кейбіреулері 5.5 кестесінде келтірілген.

5.5-кесте.

Regex класының әдістерінің кейбір опциялары

Опция	сипаттама
CultureInvariant	Предписывает игнорировать национальные установки (культуру) строки.
ExplicitCapture	Модифицирует способ поиска соответствия, обеспечивая только буквальное соответствие.
IgnoreCase	Игнорирует регистр символов во входной строке.
IgnorePattern-Whitespace	Удаляет из строки не защищенные управляющими символами пробелы и разрешает комментарии, начинающиеся со знака фунта или хеша.
Multiline	Изменяет значение символов ^ и $ так, что они применяются к началу и концу каждой строки, а не только к началу и концу всего входного текста.
RightToLeft	Предписывает читать входную строку справа налево вместо направления по умолчанию – слева на право.

5.6-кестеде кластың статистикалық әдістері келтірілген.

5.6-кесте

Regex класының статистикалық әдістері

	Escape	Преобразует минимальный набор метасимволов (\, *, +, ?, |, {, [, (,), ^, $,., # и пробел), заменяя их escape-кодами.
	IsMatch	Перегружен. Указывает на то, обнаруживает ли регулярное выражение соответствие во входной

		строке.
	Match	Перегружен. Ищет во входной строке вхождение регулярного выражения и возвращает точный результат в качестве отдельного объекта Match.
	Matches	Перегружен. Ищет во входной строке все вхождения регулярного выражения и возвращает все успешные соответствия, как если бы Match вызывался несколько раз.
	Replace	Перегружен. В указанной входной строке заменяет строки, соответствующие шаблону регулярного выражения, указанной строкой замены.
	Split	Перегружен. Разделяет входную строку в массив подстрок в позициях, определенных соответствием регулярного выражения.
	Unescape	Отменяет преобразование преобразованных символов во входной строке.

5.8 Тұрақты өрнектің элементтерін топтастыру

Тұрақты өрнектің элементтерін топтастыру үшін дөңгелек жақшалар қолданылады. Топтастыру дөңгелек жақшалардағы тұрақты өрнектің мағыналық бөлігімен сәйкес келген талданатын мәтіннің мәндерін (фрагменттерін) арнайы массивта (коллекцияда) есте сақтау үшін қолданылады.

```
      . . .
string text = " 540-356 54-03-56 ili 49-65-78.";
      . . .
foreach (Match T in Regex.Matches(text, @"(\d\d)-(\d\d)-(\d\d)"))
 {
   Console.WriteLine(T);
   Console.WriteLine(T.Groups[0]);
   Console.WriteLine(T.Groups[1]);
   Console.WriteLine(T.Groups[2]);
   Console.WriteLine(T.Groups[3]);
   Console.WriteLine(T.Groups[4]);
   Console.WriteLine(T.Groups[5]);
 }
      . . .
```

Работа программы:

54-03-56

54-03-56

54

03
56

49-65-78
49-65-78
49
65
78

Group класы Capture класының мұрагері, сонымен бірге Match класының аталығы болады. Өзінің аталығынан Index, Length және Value қаисеттерін мұраға алады және оларды өз ол ұрпақтарына береді.

«Сәйкес келуді іздестіру барысында топтар қай кезде және қалай құрастырылатынын неғұрлым толығырақ қарастырайық. Егерде тұрақты өрнекте пайдаланылатын символдарды сипаттайтын алдыңғы кестені, жекелей алғанда топтастыру символдарына назар салып талдасақ, онда топтармен байланысты бірнеше маңызды деректі анықтаймыз:

Іздестіру шартын қанағаттандыратын бір бағыныңқы жол табылғанда, бір топ емес, топтар коллекциясы құрастырылады;

индексі 0 топта табылған сәйкес келу туралы ақпарат бар;

коллекциядағы топтар саны тұрақты өрнектің жазбасындағы дөңгелек жақшалардың санына байланысты. Дөңгелек жақшалардың әр жұбы бағыныңқы жолдың дөңгелек жақшаларда берілген үлгіге сәйкес келетін бөлігін сипаттайтын қосымша топты құрайды;

топтар индекстелген болуы мүмкін, бірақ атауланған да болуы мүмкін, өйткені дөңгелек жақшаларда топтың атауын көрсетуге рұқсат етілген.

Қорытындысында, әртүрлі ақпараты бар жолдарды талдау кезінде атауланған топтарды құру өте пайдалы екенін атап кетемін.» [Основы программирования на C# Биллиг Владимир Арнольдович б.143]

Іздестіру қорытындыларының басылымынан топтардың коллекциясы екі элементтен тұратынын көреміз. Коллекция элементіндегі топтар саны тұрақты өрнектің жазбасындағы дөңгелек жақшалар жұптарының санына сәйкес келеді (индексі 0 топта табылған элемент туралы барлық ақпарат бар). Коллекция элементінің қалған топтарында топтардың дөңгелек жақшаларында берілген үлгілерге сәйкес келетін ақпарат бар.

Бағдарлама кодында индекстері 4 және 5-ке тең «қолданыста жоқ» топтардан ақпаратты шығару қосымша жазылған. Монитор экранында бос жолдар пайда болды, бірақ қателер туралы хабарламалар жоқ.

Топтастыру жеке символ үшін немесе тұрақты өрнектегі символдардың тізбегі үшін қайталағышты беру қажет болғанда пайдаланылады.

Айнымалының (қайталағыштың) атауы дөңгелек жақшаларда немесе апострофтарда беріледі, мысалы:

(?<айнымалының атауы > тұрақты өрнектің мағыналық бөлігі)

Мысалы, мәтіннен nn–nn–nn түрінде жазылған телефон нөмірлерін анықтау қажет болсын. Нөмірді іздестіру үшін тұрақты өрнекті келесідей тәсілмен жазуға болады:

```
@"(?<num>\d{2}-\d{2}-\d{2})"
```

Осы өрнекте бұрышты жақшаларда «?» символынан соң топ атауы беріледі, содан соң топтың тұрақты өрнегінің үлгісі беріледі – \d{2}-\d{2}-\d{2}. Қандай да бір мәтінде тұрақты өрнектің үлгісі бойынша телефон нөмірлерін іздестіруді атқаратын бағдарламаның фрагменті келесі түрге ие:

```
string text = " 540-356 54-03-56 ili 49-65-78.";
     . . .
foreach (Match T in Regex.Matches(text, @"(?<num>\d{2}-\d{2}-
\d{2})"))
     Console.WriteLine( T.Groups["num"]);
     . . .
```

Работа программы:

54-03-56
49-65-78

Мәтінді талдау кезінде табылған телефондардың нөмірлері Groups коллекциясына тізбектеліп жазылынатын болады. num (нөмір) айнымалысы алдын ала сипатталғанген емес және коллекцияның бөлігі – коллекция бірінші тобының атауы болып табылады. Коллекцияның әр тобының нөлдік индексін беріп, іздестіру нәтижелерін қарастыруға болады. Мысалы:

```
foreach (Match T in Regex.Matches(text, @"(?<num>\d{2}-\d{2}-
\d{2})"))
 Console.WriteLine(T.Groups[0]);
```

Работа программы:

54-03-56
49-65-78

Кері сілтемелерді қалыптастыру үшін топтастыруды пайдаланудың тағы бір нұсқасын қарастырамыз. Дөңгелек жақшаларға алынған барлық конструкциялар 1-ден бастап автоматты түрде нөмірленеді (нөлдік индекс бүкіл тұрақты өрнекті белгілеу үшіне қолданылады). Алдын ала кері қисық сызықпен белгіленген осы нөмірлерді сәйкес конструкцияға сілтеме үшін пайдалануға болады. Мысалы, (\w)\1 өрнегі, нақты айтқанда @”(\w)\1”, бірдей әліпби-цифрлық символдардың барлық жұптарын табуға мүкіндік береді. Әдетте осындай өрнек сөздерде қосарланған символдарды іздестіру үшін пайдаланылады, мысалы, class, mass.

```
. . .
string text = "Язык регуляррrных выражжений опредееляет шшаблоны
символов.";
     . . .
foreach (Match T in Regex.Matches(text,@"(\w)\1"))
  Console.WriteLine("{0} - {1}", T.Index, T.Value);
     . . .
```

Работа программы:
12 - рр
23 - жж
35 - ее
42 - шш

Атауы бұрышты жақшалардағы өрнекте берілетін айнымалыны сондай-ақ өрнектің кейінгі бөлігіне кері сілтемелер үшін қолдануға болады. Мысалы, сөздердегі қосарлы символдарды іздестіруді @”((?<s>\w)\k<s>)” өрнегінің көмегімен атқаруға болады. Осы тұрақты өрнекі қарастырайық. Топ «(?<s>\w)» өрнегімен берілген. Бұрышты жақшалардағы сұрақ белгісінен кейін «<s>» топтың атауы берілген. Топ тауынан кейін осы топты сипаттайтын үлгі орналасады, біздің мысалда үлгі «\w» өрнегімен беріледі. Осы өрнек топ бір кез келген әліпби-цифрлық символдан, яғни кіші және үлкен әріптердің және ондық цифрлардың жиынтығындағы символдан тұратынын білдіреді. Топтың сипатталуынан соң кері сілтеменің белгісі «\k» символдарының жұбы орналасқан. Осы жұптан соң «<s>» деген топ атауы орналасқан.

```
. . .
string text = "Язык регуляррррных выражжений определяет шаблоны символов.";
     . . .
foreach (Match T in Regex.Matches(text, @"((?<s>\w)\k<s>)"))
 {
   Console.WriteLine(T.Groups["s"]);
   Console.WriteLine(T.Groups[0]);
 }
     . . .
```

Работа программы:
р
рр
ж
жж
е
ее
ш
шш

«Топ атауы» үшін және нөлдік индексті топ үшін іздестіру нәтижелерін шығару әртүрлі. Топ атауына тек бір символ – үлгі сәйкес келсе, нөлдік индексті топқа табылғанның барлығы сәйкес келеді.

Символ регистрін ескермей іздестіруді ұйымдастыруға болады:

```
. . .
string text = "Язык регуляррррных выражжений опредееляет шшаблоны символов.";
     . . .
foreach (Match T in Regex.Matches(text,
```

```
                    @"((?<s>\w)\k<s>)", RegexOptions.IgnoreCase)
 {
   Console.WriteLine(T.Groups["s"]);
   Console.WriteLine(T.Groups[0]);
 }
     . . .
```

Работа программы:

Я
Яя
р
рр
ж
жж
е
ее
ш
шш

Мәтінде қатарынан орналасқан және пробелдардың ерікті санымен бөлінген қайталана беретін сөздерді іздестіру үшін (регистріне қарамастан) тұрақты өрнектің келесі конструкторын пайдалануға болады:

```
. . .
Regex reg = new Regex(@"\b(?<word>\w+)\s+(\k<word>)\b",
RegexOptions.IgnoreCase);
string text = "Class   class Regex это Это class regex регулярных
выражений.";
     . . .
foreach (Match T in reg.Matches(text))
 {
   Console.WriteLine(T.Groups["word"]);
   Console.WriteLine(T.Groups[0]);
 }
     . . .
```

Работа программы:

Class
Class class
это
это Это

Мәтінде жолдың кез келген позициясында орналасқан қайталана беретін сөздерді (регистрге қарамастан) іздестіру үшін, « бөлгіш пробелдарды– \s» кез келген символдарға – «.» ауыстыру керек.

```
. . .
Regex reg = new Regex(@"\b(?<word>\w+).+(\k<word>)\b",
RegexOptions.IgnoreCase);
string text = "Class   class Regex это Это class regex регулярных
выражений.";
     . . .
foreach (Match T in reg.Matches(text))
```

```
{
  Console.WriteLine(T.Groups["word"]);
  Console.WriteLine(T.Groups[0]);
}
    . . .
```

Работа программы:
Class
Class class Regex это Это class

Іздестіру нәтижесі – олар көп болу керек сияқты болса да, ол тек бір топ. «.+» қайталағышының (оның жиі жағдайда тойымсыз деп атайды «greedy») жұмысына назар аударайық, ол алдыңғы элементтің бір немесе одан да көп қайталануын білдіреді. Мәтіннің бірінші сөзі үшін, ал ол «Class» сөзі, қайталағыш мәтіннің соңына дейін барлық сәйкес келулерді іздестіреді де және табылған фрагмент мәтіннен «кесілініп алынады» да кейінгі талдауға қатыспайды.
Мысалымыздың мәтінін келесі түрде өзгертейік:

```
string text = "Class Это class Regex это Это  regex это регулярные Это  выражения.";
```

Работа программы:
Class
Class Это class
Regex
Regex это Это regex
это
это регулярные Это

Іздестіру нәтижесінде коллекцияның үш тобы табылды.

5.9 Тұрақты өрнектерде Split және Replace әдістерін пайдалану.

Regex класында мәтінде іздестіру әдістерімен қатар, мәтінді өңдеудің кейбір басқа әдістері де бар. Regex класының іздестіру жүргізбейтін әдістерінің арасында Split әдісі жиі қолданылады. Ол тұрақты өрнектерге мәтінді бөлшектегішті беруге мүмкіндік береді. Екінші жиі қолданылатын әдіс Replace, ол берілген тұрақты өрнекке сәйкес мәтін фрагменттерінің ауыстыруын атқарады.

Оқу мысалдарында осы әдістердің пайдаланылуын қарастырамыз.

Осы тіліндегі қандай да бір мәтіні кодтау үшін 00-ден 32-ге дейінгі (регистрдің мәні есепке алынбайды) ондық сандар қолданылады деп болжаймыз. «Үтір» символы 33 санымен, «нүкте» –34 санымен, ал «пробел» символы 35 санымен кодталады. Мәтіннің қалған символдары, яғни цифрлармен басқа әліпбилердің әріптері 40-тан 99-ға дейінгі диапазонда кездейсоқ сандармен кодталады. Кодталған мәтінде сандар «пробел» символымен бөлінеді. Алғашқы мәтінді келтірілген ережелердің көмегімен кодтап, оны жаңа уақытша мәтінге

жазу қажет. Уақытша мәтіннің декодтауын қарастыру қажет. Декодтау 00-ден 35-ке дейінгі сандарды түрлендіруді келтірілген ережелерге сәйкес орындау керек. 40-тан 99-ға дейінгі диапазондағы сандарды елемеу керек. Монитор экранында алғашқы, кодталған және декодталған мәтіндерді көруді қарастыру қажет. Бағдарламада Regex класының Split және Replace әдісін пайдалану керек.

«PreobText» деген жеке клас құрамыз, онда конструкторды және мәтінді кодтау және декодтаудың жолдық әдістерін ұйымдастырамыз. Мәтіннің өзі конструкторға клас өрістерінің инициализациялануы кезінде берілетін болады.

Мәтінді кодтау әдісіне Regex класының Split әдісінің көмегімен алғашқы мәтінді «сөздерге» «бөлу» кіреді. Әрі қарай тапсырмаға сәйкес мәтіннің әр «сөзінің» кодталауы орындалады. Алынған екі разрядты сандардың тізбектері жаңа мәтінге «біріктіріледі», ол әдіспен бағдарламаға қайтарылады.

Декодтау әдісі екі разрядты сандардың тізбегін Regex класының Replace әдісін пайдалана отырып жаңа мәтінге түрлендіреді. Мәтін әрі қарай тапсырмаға сәйкес декодталады.

Бағдарлама жұмысының әр кезеңінде (бағдарламаның менюі) меню пункті жұмысының нәтижесін басу қарастырылған.

Бағдарлама коды:

```
using System;
using System.Collections.Generic;
using System.Text.RegularExpressions;
using System.Text;

namespace ConsoleApplication1
{
 class Program
 {
  class PreobText
  {
   string text;
   public string rez;
   string alfavit = "абвгдеёжэийклмнопрстуфхцчшщъыьэюя,. ";

   //Конструктор
   public PreobText(string txt)
   {
    text = txt;
    rez = "";
   }
   // метод кодирования
   public string Codirovanie()
   {
    bool ok = false;
    text = text.ToLower();
    int m = 0, k = 0, Kol = 0, r = 0;
    Random rnd = new Random();
    string p, clo;
    //Console.WriteLine("nacalo preobrazovanij!");
    //Console.WriteLine(text);
    //Console.WriteLine(rez);
```

```
List<string> clova=new List<string>(Regex.Split(text,@"[ ]"));
   m = clova.Count;
   string[] newclova = new string[m];
   foreach (string ss in clova)
   {
    Kol = ss.Length;
    //  Console.WriteLine(Kol);
    clo = "";
    if (Kol != 0)
    {
     for (int j = 0; j < Kol; j++)
     {
      ok = false;
      for (int i = 0; i < 36; i++)
      {
       if (ss[j] == alfavit[i])
       {
        ok = true;
        if (i <= 9)
        { p = "0" + i.ToString() + " "; }
          else p = i.ToString() + " ";
        clo = clo + p;
       }
      }
      if (ok == false)
      {
       r = rnd.Next(40, 99);
       clo = clo + r.ToString()+ " ";
      }
     }
     clo.Trim();
     if (clo != "") { newclova[k] = clo; k++; }
     //Console.WriteLine(newclova[k - 1]);
    }
   }

   // Console.WriteLine("konec preobrazovanij!");
   // Console.WriteLine();
   for (int j = 0; j < m; j++)
   {
    //Console.WriteLine(newclova[j]);
    rez = rez + newclova[j] + "35" + " ";
   };
   return rez;
  }

  // метод декодирования
  public string Decodirovanie()
  {
   string dec = "";
   int i=0,Kol=0;
   dec = Regex.Replace(rez, @"\d{2}", @"<$0");
   Console.WriteLine(dec);
   List<string> clova=new List<string>(Regex.Split(dec,@"<"));
```

```
    Kol = clova.Count;
    // Console.WriteLine(Kol);
    rez = "";
    foreach (string ss in clova)
    {
     Kol = ss.Length;
     if (Kol != 0)
     {
      i = Convert.ToInt32(ss);
      if (i < 36) rez = rez + alfavit[i];
     }
    }
    // Console.WriteLine("Konec");
    return rez;
   }
  }

  static void Main(string[] args)
  {
   string txt = "CG1но34в55а! н41ас веGamesдуZ23т куТід44а 8767 то,
с62нкwовtta я таскaлщу росюакотзукак.";
   PreobText ob = new PreobText(txt);
   string ntxt;
   int w = 0;
   string buf;
   while (w < 4)
   {
    Console.WriteLine("1 - Просмотр исходного текста");
    Console.WriteLine("2 - Кодирование текста");
    Console.WriteLine("3 - Декодирование текста");
    Console.WriteLine("4 - Выход из программы");
    Console.WriteLine("Введите пункт меню программы");
    buf = Console.ReadLine();
    w = Convert.ToInt32(buf);
    switch (w)
    {
     case 1: Console.WriteLine(txt); break;
     case 2:
            {
             ntxt = ob.Codirovanie();
             Console.WriteLine(ntxt);
             break;
            }
     case 3:
            {
             ntxt = ob.Decodirovanie();
             Console.WriteLine(ntxt);
             break;
            }
     default: break;
    }
   }
  }
 }
```

```
}
```
Работа программы:
1 - Просмотр исходного текста
2 - Кодирование текста
3 - Декодирование текста
4 - Выход из программы
Введите пункт меню программы
1
CG1но34в55а! н41ас веGamesдуZ23т куTід44а 8767 то, с62нkwовtta я tackaлщу росюак
omзykак.
1 - Просмотр исходного текста
2 - Кодирование текста
3 - Декодирование текста
4 - Выход из программы
Введите пункт меню программы
2
18 41 79 14 15 50 87 02 80 59 00 53 35 14 59 60 00 18 35 02 05 70 54 88 77 47 04 20 43 47 77 19 35 11 20 55 98 04 46 52 00 35 46 70 97 76 35 19 15 33 35 18 54 90 14 68 79 15 02 54 77 00 35 32 35 19 00 85 86 62 56 26 20 35 17 91 54 31 98 11 74 79 72 92 45 00 11 34 35
1 - Просмотр исходного текста
2 - Кодирование текста
3 - Декодирование текста
4 - Выход из программы
Введите пункт меню программы
3
<18 <41 <79 <14 <15 <50 <87 <02 <80 <59 <00 <53 <35 <14 <59 <60 <00 <18 <35 <02
<05 <70 <54 <88 <77 <47 <04 <20 <43 <47 <77 <19 <35 <11 <20 <55 <98 <04 <46 <52
<00 <35 <46 <70 <97 <76 <35 <19 <15 <33 <35 <18 <54 <90 <14 <68 <79 <15 <02 <54
<77 <00 <35 <32 <35 <19 <00 <85 <86 <62 <56 <26 <20 <35 <17 <91 <54 <31 <98 <11
<74 <79 <72 <92 <45 <00 <11 <34 <35
снова нас ведут куда то, снова я тащу рюкзак.
1 - Просмотр исходного текста
2 - Кодирование текста
3 - Декодирование текста
4 - Выход из программы
Введите пункт меню программы