ЛЕКЦИЯ 7

СЪДЪРЖАНИЕ

I. РЕГУЛЯРНИ ИЗРАЗИ – СЪЩНОСТ, ПРОИЗХОД И ОСНОВНИ ПРИЛОЖЕНИЯ

Какво е регулярен израз?

Регулярните изрази представляват мощен апарат **за обработка на символни низове**.

Регулярният израз е символен низ, конструиран по специални синтактични правила. Той описва един език от думи – символни низове.

На регулярния израз може да се гледа като на шаблон – той търси съвпадения с всички думи, които отговарят на този шаблон. Един от най-често използваните регулярни изрази, например, е изразът:

```
(.+)@(.+)\.(.+)
```

Макар да изглежда странно на пръв поглед, това не е нищо друго освен един опростен шаблон за формата на стандартните e-mail адреси, с които всеки е запознат – поредица от символи, следвана от символа @ и после име на домейн.

I. РЕГУЛЯРНИ ИЗРАЗИ – СЪЩНОСТ, ПРОИЗХОД И ОСНОВНИ ПРИЛОЖЕНИЯ

За какво се използват регулярните изрази?

Регулярните изрази могат да се използват за най-разнообразни задачи при текстообработката. В практиката те най-често са полезни при три типа задачи:

- **Търсене в** текст и **извличане** на полезна информация от текста;

- **Валидация** на входни данни. Валидацията на входни данни е задължителна във всеки съвременен софтуер, който претендира да спазва елементарните изисквания за сигурност.

- **Заместване**. Заместването на един низ с друг в текст е една от най-често срещаните практически задачи при текстообработката, а същевременно и доста бавна и трудна за реализация. С помощта на регулярните изрази подобен проблем се разрешава само с няколко реда код.

Например можем да намерим всички срещания на конструкции от типа "(a+b)*c" и да ги заместим с еквивалентните им "a*c+b*c" – нещо, което е доста трудно за постигане с традиционните средства за синтактичен анализ и текстообработка.

I. РЕГУЛЯРНИ ИЗРАЗИ – СЪЩНОСТ, ПРОИЗХОД И ОСНОВНИ ПРИЛОЖЕНИЯ

Няколко прости примера за регулярни изрази и тяхното използване

Регулярните изрази сами по себе си са низове, които използваме, за да търсим съвпадения на символни последователности с дефинирания шаблон.

Пример 1:

```
пример номер [0-9]+
```

Този шаблон ще намери съвпадения с всички поднизове, които започват с фразата "пример номер", следвана от поне една десетична цифра.

Квадратните скоби в езика на регулярните изрази указват клас от символи – в случая това са всички символи в диапазона между 0 и 9. Знакът + ни казва, че предходният символ (или в случая целият клас символи [0-9]) трябва да се среща последователно 1 или повече пъти. Така низът "пример номер 987" ще бъде счетен за съвпадение с шаблона, както и низът "пример номер 05". Низът "пример номер" няма да е съвпадение, защото по шаблона трябва да има поне една цифра на края.

I. РЕГУЛЯРНИ ИЗРАЗИ – СЪЩНОСТ, ПРОИЗХОД И ОСНОВНИ ПРИЛОЖЕНИЯ

Пример 2: *Търсене на телефонни номера*

```
(+359|0)88[0-9]{7}
```

С този израз можем да намерим съвпадения с номера на мобилни телефони в мрежата на А1, например. Отново използваме символния клас за цифри [0-9], следван от модификатора за количество {7} – това означава, че трябва да има точно 7 повторения на символи от класа [0-9]. В началото имаме (+359|0), което означава, че номерът трябва да започва или с +359, или с 0. Символът | обозначава алтернативни възможности.

Примери за номера, който съвпадат с шаблона, са "+359887675340" и "0888997621", но не и "188385953" (започва с 1), или "+3598890076" (има само пет цифри след "+35988").

II. РЕГУЛЯРНИ ИЗРАЗИ В .NET ПЛАТФОРМАТА

Пример за регулярен израз, реализиран с код в .NET

```
static void Main()
{
   string text = "Няма скара, няма бира, няма какво да ям";
   string pattern = @"\w*ира|скара|\w*ям\w*";
   Match match = Regex.Match(text, pattern);
   while (match.Success)
   {
      Console.WriteLine(
         "Низ: \"{0}\" - начало {1}, дължина {2}",
         match, match.Index, match.Length);
      match = match.NextMatch();
   }
}
```

Низ, в който ще търсим поднизове

Шаблон, по който ще се търси

Метод за търсене на съвпадение

```
Низ: "Няма" - начало 0, дължина 4
Низ: "скара" - начало 5, дължина 5
Низ: "няма" - начало 12, дължина 4
Низ: "бира" - начало 17, дължина 4
Низ: "няма" - начало 23, дължина 4
Низ: "ям" - начало 36, дължина 2
```

Как работи примерът?

Регулярният израз, който се съдържа в низа pattern, търси за три алтернативни възможности:

- подниз, завършващ на "ира" – това се сочи от частта "\w*ира", където \w* означава произволна последователност (може и с нулева дължина) от букви, цифри и знак за подчертаване.
- подниз "скара".
- подниз (отново от букви, цифри и знак за подчертаване), който съдържа подниза "ям".

За търсенето на съвпадения се използват методите на класовете **Regex (Regular Expressions)** и **Match**.

III. ЕЗИКЪТ НА РЕГУЛЯРНИТЕ ИЗРАЗИ

Богатството на синтаксиса на регулярните изрази позволява голяма гъвкавост при изграждането на шаблони. Пълните възможности на езика рядко се ползват в практиката, но за по-специфични задачи все пак може да се наложи и тяхното познаване.

В синтактичен план езикът на регулярните изрази се дели на две основни групи – **литерали** и **метасимволи** (символи със специално значение).

Литерали

Литералите се използват, за да фиксираме частта от шаблона, която задължително трябва да присъства точно в посочения вид. Например в регулярния израз, съхраняван в променливата pattern, която разгледахме в примера по-горе, частите "**ира**", "**скара**" и "**ям**" са **литерали** – те се търсят в текста от машината на регулярните изрази (regex engine) точно в този си вид, в който ги виждаме написани.

III. ЕЗИКЪТ НА РЕГУЛЯРНИТЕ ИЗРАЗИ

Метасимволи

Метасимволите са група от символи, които имат специално значение в езика на регулярните изрази. С тяхна помощ реално се постига цялата функционалност и описателност на апарата, който разглеждаме. Метасимволите, с които разполагаме, осигуряват достатъчно гъвкавост, за да можем да опишем практически почти всеки шаблон, който би могъл да ни трябва.

Символите, които се третират по специален начин от машината на регулярните изрази в зависимост от позицията си, са следните**: [,], |, *, . , +, ?, (,), {, }**, \, ^ **и $.** Най-често обаче терминът "метасимволи" се използва не за тях, а за специалните конструкции, които носят определен смисъл за шаблона и в които тези символи участват.

В примера, който разгледахме по-горе, метасимволи са " **\w** " и " ***** ", както и " **|** ".

Escaping при регулярните изрази

Тъй като метасимволите се третират по нетривиален начин при парсването на регулярния израз, те не могат да се използват като литерали директно и не могат да се търсят като обикновени символи. Както и при повечето програмни езици, този проблем се избягва с помощта на escaping чрез символа \ **(backslash).** Поставен пред някой от специалните символи, които изброихме по-горе, този символ отменя особеното им значение и те се третират по обикновения начин. Например * означава *, \+ означава + и т.н. Това важи разбира се и за самия символ \, т.е. ако искаме да го използваме като литерал, трябва да използваме \\.

Някои от изброените по-горе метасимволи имат специално значение само на определени места в шаблона. Например символът], както и символите) и } са метасимволи само ако преди тях в шаблона има незатворена съответната отваряща скоба. Ако се опитаме да ползваме escaping на тези символи в друг контекст, ще получим грешка, защото това обърква парсването.

III. ЕЗИКЪТ НА РЕГУЛЯРНИТЕ ИЗРАЗИ

Други escaping последователности

\n – нов ред (new line)
\r – връщане на каретката (carriage return)
\t – табулация (tab)
\a – звуков сигнал (bell)
\b – връщане назад (backspace)

Комбинацията **\b**, има по-специално значение за синтаксиса на регулярните изрази. Ето защо можем да я използваме **като backspace** единствено в споменатата конструкция за класът символи [ ] и в шаблоните за заместване.

Някои особености

Когато използваме escaping при регулярните изрази в .NET Framework, не трябва да забравяме, че се налага да се съобразяваме и с компилатора. Преди да се подадат на машината на регулярните изрази, низовете, които използваме за шаблони, първо се обработват от парсер, който има собствени escaping последователности.

В езика C# символът \ се използва в низовете също за escaping. Трябва да се съобразяваме с този факт, защото можем да стигнем до нежелани резултати, ако не внимаваме. Да разгледаме шаблона "\w*", най-общо означава произволна дума. Ако напишем този низ в нашия сорс код, ще получим грешка при компилация, защото компилаторът на C# не може да разпознае "\w" като коректна escaping последователност. Ето защо трябва да освободим символа \ от специалното му значение, като използваме escaping и за него – "\\w*". Сега вече след преминаването през парсера на C# компилатора този низ ще има вида "\w*", което е коректно за машината за регулярни изрази, която ще го обработи по време на изпълнение.

Какво трябва да направим, ако искаме да използваме \ като обикновен символ в израза (когато търсим последователност, в която той участва)?

Тогава трябва да имаме "\\" в шаблона, например "c:\\windows". Но преди да се разчете като шаблон от машината за регулярни изрази, този низ ще мине през парсера по време на компилация и там двата символа backslash ще се редуцират до един по правилата за escaping в C#, при което ще получим в резултат "c:\windows". Тук обаче \w има значение на метасимвол и ще получим нежелан резултат. Затова в сорс кода трябва да имаме "c:\\\\windows" – т.е. четири повторения на \, за да намерим съвпадение с един такъв символ.

Използване на символа @

В езика C# при поставянето на @ пред отварящите кавички на низа всички специални символи в него (освен кавичките) губят специалното си значение. Това е много удобно при шаблоните за регулярни изрази и е препоръчително да се използва винаги, защото спестява проблемите, описани в предходните два слайда.

```
string patternWord = @"\w+";
//Same as: patternWord = "\\w+";
string patternDir = @"c:\\windows";
//Same as: patternDir = "c:\\\\windows";
```

Използването на @ ни позволява да записваме низа във вида, в който той ще се прочете от машината за регулярните изрази (с малки изключения за кавичките). Това е доста удобно и е добре да го използваме.

III. ЕЗИКЪТ НА РЕГУЛЯРНИТЕ ИЗРАЗИ

Най-важното за работата с регулярни изрази !

1) Шаблонът не е за съвпадение с целия низ - Когато търсим съвпадение по даден шаблон, ние търсим произволен подниз на нашия низ, който да отговаря на шаблона. В текста може да има много такива поднизове. Шаблонът не описва целия текст (освен ако изрично не укажем това), а описва поредица символи, която се търси в текста. Например шаблонът "**бира**" има две съвпадения в низа "Разбирам от бира" – едното е част от думата "разбирам", а другото е самата дума "бира".

2) Съвпаденията се откриват в реда на срещане - Регулярните изрази винаги работят отляво надясно по низа и откриват първо най-левите съвпадения. В горния пример поднизът "**бира**" в думата "Разбирам" ще бъде открит пръв, макар че ние вероятно сме искали да търсим за самата дума "бира". Това е често срещан проблем и по-нататък в темата ще се научим как да се справяме с него.

3) Търсенето приключва, когато се открие съвпадение - По принцип целта на търсенето с регулярни изрази е да се открие някакво съвпадение с шаблона. Машината на регулярните изрази се стреми да не хаби излишни ресурси и прекратява хода си в момента, в който се открие съвпадение. Ако полученото съвпадение не ни устройва, можем да търсим отново или да използваме подходящите методи за събиране на всички съвпадения до пълното изчерпване на низа наведнъж.

4) Търсенето продължава от последното съвпадение - Ако сме намерили съвпадение с регулярен израз и след това подновим търсенето в същия текст със същия регулярен израз, то продължава от мястото след последното намерено съвпадение. Това е важна подробност, която трябва добре да се запомни, защото често е причина за объркване! За да я изясним, ще разгледаме примерния шаблон "(бира!){2}". С този шаблон търсим две последователни повторения на подниза "бира!", тоест на практика търсим "бира!бира!". Нека да го приложим към примерния низ "бира!бира!бира!". В текста реално има две срещания на нашия шаблон – едното започва от позиция 0 ("**бира!бира!**бира!"), а другото – от позиция 5 ("бира!**бира!бира!**").

Основни метасимволи

Ще разгледаме най-често срещаните метасимволи в езика на регулярните изрази

1) Класове от символи - Това са метасимволи, които означават цяло множество от обикновени символи. Срещането на който и да е символ от множеството (или класа) на съответното място в низа се счита за съвпадение. Използват се следните означения за различните видове класове от символи:

- **Точка** - Специалният символ **.** обозначава класа от всички символи, с изключение на символа \n за нов ред. Като пример за този метасимвол, да разгледаме шаблона **"би.а"** и няколко низа, към които да го приложим:

```
Шаблон: би.а
бира – има съвпадение "бира"
бива – има съвпадение "бива"
би+а – има съвпадение "би+а"
бивна – няма съвпадение (между "би" и "а" има два символа, а не един)
```

III. ЕЗИКЪТ НА РЕГУЛЯРНИТЕ ИЗРАЗИ

- **Квадратни скоби (класове с изброяване)** - Конструкцията **[**редица_символи**]** е метасимвол, обозначаващ класа от всички обикновени символи, изброени в редицата. Символите се изброяват без никакви разделители между тях. Тук има няколко варианта. Единият е простото изброяване на символи от вида [символи]. Това е стандартният вид на конструкцията, който вече обяснихме:

```
Шаблон: би[вр]а
разбира – има съвпадение "бира"
не бива – има съвпадение "бива"
биха ни – няма съвпадение (между "би" и "а" трябва "р" или "в")
```

Обърнете внимание, че редът на символите в квадратните скоби няма значение за реда на намерените съвпадения – важен е редът на срещане в низа. Съвпадението с "бира" е първо в "Тази бира я бива", въпреки че в шаблона "в" стои преди "р" в квадратните скоби.

III. ЕЗИКЪТ НА РЕГУЛЯРНИТЕ ИЗРАЗИ

- **Отрицание на клас** - Това е конструкцията [^символи]. Символът ^, поставен веднага след отварящата квадратна скоба, означава, че се търсят съвпадения с всички символи, които НЕ влизат в класа на изброяваните. Важно е да се запомни, че символът ^ има специално значение, единствено когато е поставен веднага след отварящата квадратна скоба, иначе се третира като обикновен символ:

```
Шаблон: би[^вр]а
разбира – няма съвпадение (между "би" и "а" не трябва да има "р" и "в")
биха ни – има съвпадение "биха"

Шаблон: би[в^р]а
разбира – има съвпадение "бира"
биха ни – няма съвпадение (между "би" и "а" трябва "р", "^" или "в")
би^а – има съвпадение "би^а"
```

III. ЕЗИКЪТ НА РЕГУЛЯРНИТЕ ИЗРАЗИ

- **Изброяване с диапазон** - Можем да използваме и следния вид клас от символи: [символА-символВ]. Този вариант на конструкцията с квадратните скоби търси съвпадения с всички символи, намиращи се в затворения интервал между символА и символВ в кодовата таблица. Подобно на ‘ ^ ‘ символът ‘ – ’ има специално значение, само ако е между други символи, а не в началото и в края.

```
Шаблон: би[б-п]а
разбира – няма съвпадение ("р" не е в интервала "б-п")
не бива – има съвпадение "бива"

Шаблон: би[-бп]а
не бива – няма съвпадение (между "би" и "а" трябва "б", "п" или "-")
би-а – има съвпадение "би-а"
```

Трите вида изброяване могат свободно да се комбинират – например "[^a-zA-Z01]" намира съвпадение с всеки символ, който не е малка или голяма латинска буква, нито цифрите 0 или 1.

2) Метасимволи за количество

Едни от най-често използваните метасимволи са тези за количество повторения на даден подниз или символ в шаблона

- **Символът * – нула или повече повторения** - С * означаваме 0 или повече повторения на символа (или метасимвол, включително клас от символи), предхождащ знака *:

```
Шаблон: бира*
разбирам – има съвпадение "бира"
бираааааа! – има съвпадение "бираааааа"
бирррррра! – има съвпадение "бир" (* важи само за символа "а")
бирено коремче – има съвпадение "бир" (0 повторения на "а")
```

Ако преди звездичката има клас от символи, то търсим поредица от последователни срещания на символи от този клас. Не се изисква обаче това да са повторения на един и същи символ от този клас.

звездичката вътре в квадратните скоби губи специалното си значение:

```
Шаблон: ба[ла]*йка
балалайка – има съвпадение "балалайка"
баллалаалайка – има съвпадение "баллалаалайка"
баалайка – има съвпадение "баалайка"

Шаблон: ба[л*а*]йка
балалайка – няма съвпадение (трябва "л", "а" или "*" по средата)
байка – няма съвпадение
ба*йка – има съвпадение "байка"
```

Горният пример показва, че не можем да използваме квадратните скоби и звездичката, за да означим повторение на някой от символите в класа на квадратните скоби. Ако искаме няколко пъти "а" или няколко пъти "б", не ни върши работа нито "[а*б*]" (търси само веднъж "а", "б" или "*"), нито "[аб]*" (което пък намира например "аббаб").

Символът + – едно или повече повторения

Този метасимвол е идентичен със символа *, като единствената разлика между двата, е, че + изисква задължително поне едно повторение на конструкцията, за която се отнася – т.е. той съвпада 1 или повече повторения.

```
Шаблон: бира+
бираааааа! – има съвпадение "бираааааа"
бирено коремче – няма съвпадение (трябва поне едно "а" след "бир")
```

Символът ? – нула или едно повторения - означава 0 или 1 повторения. Обикновено се използва за някаква незадължителна конструкция в шаб-лона, която или се среща само веднъж, или изобщо не се среща. В примера отново използваме ограждането със скоби за цяла група.

```
Шаблон: няма( бира)?
няма бира – има съвпадение "няма бира"
бира няма – има съвпадение "няма"
```

Метасимволи за точен брой повторения

Ако искаме да укажем с по-голяма точност броя на последователните срещания на даден символ или конструкция, можем да използваме специалните символи за точен брой повторения. С {n} указваме, че предхождащият (мета)символ ще се повтаря точно **n** пъти. {n,} означава поне **n** повторения, а {**n,m**} е за поне n, но не повече от m последователни срещания на дадената конструкция:

```
Шаблон: бир{2,3}а*
разбирам – няма съвпадение ("р" трябва да се среща поне 2 пъти)
бирраааааа! – има съвпадение "бирраааааа"
бирррррра! – има съвпадение "бирр" ("р" се среща 4 пъти,
конструкцията позволява до 3 и взима максималния брой)
```

Лесно можем да забележим, че познатите ни *, + и ? могат да се представят чрез символите за точен брой повторения, както следва: * като {0,}, + като {1,} и ? като {0,1}.

3) Метасимволи за местоположение

Метасимволите за местоположение се различават от вече разгледаните типове, защото не се използват за съвпадане на символи в текста, а за съвпадане с позиция в текста.

- **Символът ^ -** С помощта на този символ можем да поискаме съвпадението да се намира в началото на низа. Символът ^ е шаблон за позицията преди първия символ.

```
Шаблон: ^бира
разбирам – няма съвпадение (поднизът "бира" не е в началото)
бираааааа! – има съвпадение "бира"
бирено коремче – няма съвпадение (няма "бира" в началото)
```

III. ЕЗИКЪТ НА РЕГУЛЯРНИТЕ ИЗРАЗИ

- **Символът $** - символът $ изисква съвпадение с края на низа (позицията след последния символ). Има едно изключение от това правило и то е ако низът завършва със символа за нов ред \n. Тогава $ намира съвпадение и с позицията преди този символ:

Шаблон: **бира$**
разбирам – няма съвпадение (поднизът "бира" не е в края)
хубава бира – има съвпадение "бира"
скарата гълта много бира\n – пак има съвпадение "бира" (има само "\n" до края)

III. ЕЗИКЪТ НА РЕГУЛЯРНИТЕ ИЗРАЗИ

- **Символите ^ и $ в многоредов режим** - Често когато четем например текст от файл, където има много нови редове, се интересуваме от позицията на търсения низ в рамките на реда, а не на целия текст. Синтаксисът на регулярните изрази позволява символите ^ и $ да означават съответно началната и крайната позиция не само на целия низ, в който търсим, а и на даден ред от него.

```
Шаблон: ^бира
лято е.\n
бирата е студена – има съвпадение "бира"
лято е,\n
но бирата е студена – няма съвпадение ("бира" не е в начало на ред)
```

III. ЕЗИКЪТ НА РЕГУЛЯРНИТЕ ИЗРАЗИ

- **Символите \A, \Z и \z -** Символите \A и \Z напълно отговарят по функционалност съответно на символите ^ и $, с тази разлика, че не позволяват многоредовото съвпадане. Те винаги откриват за съвпадение единствено началото и края на целия низ, а не на всеки ред. Символът \z се различава от \Z и от $ по това, че той не открива съвпадение преди "\n" на края на низа:

```
Шаблон: бира\Z
скарата гълта много бира\n – има съвпадение "бира"

Шаблон: бира\z
скарата гълта много бира\n – няма съвпадение
```

4) Метасимволът за избор |

Конструкция от вида "**regexp1 | regexp2**" означава, че успешно съвпадение ще бъде открито както ако низът отговаря на шаблона "regexp1", така и ако отговаря на шаблона "regexp2". На практика с метасимвола | даваме на машината за регулярни изрази възможност за избор. Ако намери съвпадение с израза от лявата страна, шаблонът е удовлетворен. Ако това не стане, машината опитва да намери съвпадение с израза от дясната страна, с което може и да успее.

Шаблон: **бира|биричка**
хубава бира – има съвпадение "бира"
биричката става – има съвпадение "биричка"
бирено коремче – няма съвпадение
бирбиричка – няма съвпадение (| не е само между "а" и "б")

Шаблон: **вишн(а|я)та**
вишната – има съвпадение "вишната"
ята – няма съвпадение (| е само между "а" и "я")