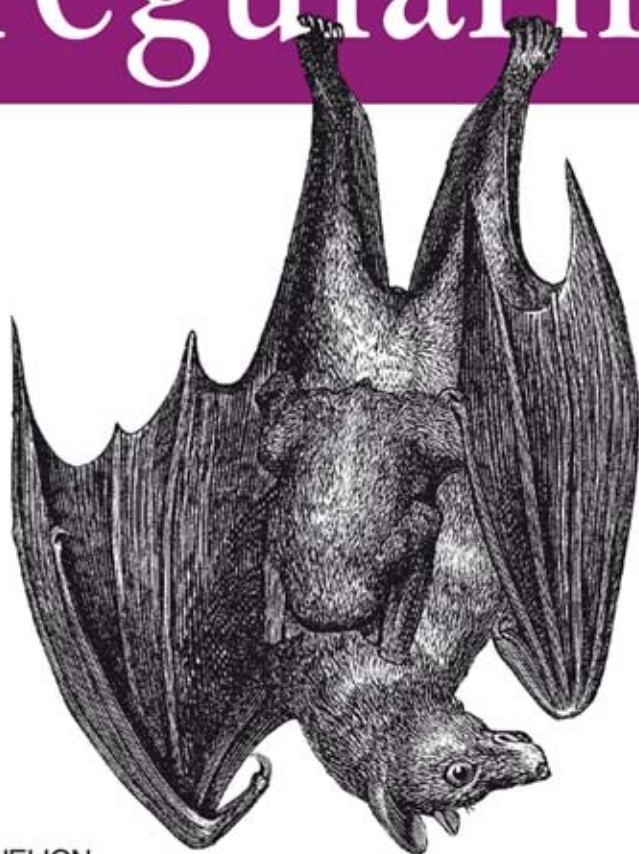


Odkryj moc wyrażeń regularnych!

Wprowadzenie

Wyrażenia regularne



HELION

O'REILLY®

Michael Fitzgerald

Tytuł oryginału: Introducing Regular Expressions

Tłumaczenie: Robert Górczyński

ISBN: 978-83-246-6868-7

© 2013 Helion S.A.

Authorized Polish translation of the English edition Introducing Regular Expressions, ISBN 9781449392680 © Michael Fitzgerald.

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Wydawnictwo HELION dołożyło wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie bierze jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Wydawnictwo HELION nie ponosi również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/wyrawp>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Wprowadzenie	7
1. Czym są wyrażenia regularne?	13
Poznaj aplikację RegexPal	14
Dopasowanie numeru telefonu w formacie stosowanym w Ameryce Północnej	16
Dopasowanie cyfr za pomocą klasy znaków	17
Używanie znaków skrótów	18
Dopasowanie dowolnego znaku	19
Grupy przechwytywania i odwołania wsteczne	19
Używanie kwantyfikatorów	20
Używanie dosłownych znaków	21
Przykłady aplikacji	23
Czego dowiedziałeś się z rozdziału 1.?	25
Informacje techniczne	26
2. Proste dopasowanie wzorca	27
Dopasowanie dosłownego ciągu tekstowego	29
Dopasowanie cyfr	30
Dopasowanie znaków innych niż cyfry	32
Dopasowanie słów i znaków niebędących słowami	32
Dopasowanie znaku niewidocznego	35
Dopasowanie dowolnego znaku, po raz kolejny	37

Oznaczanie tekstu	39
Czego dowiedziałeś się z rozdziału 2.?	43
Informacje techniczne	44
3. Granice	47
Początek i koniec wiersza	47
Granice słowa i niesłowa	49
Inne kotwice	52
Określenie grupy znaków jako dosłownych	53
Dodawanie znaczników	54
Czego dowiedziałeś się z rozdziału 3.?	58
Informacje techniczne	59
4. Alternatywy, grupy i odniesienia	61
Alternatywy	62
Podwzorce	65
Grupy przechwytywania i odwołania wsteczne	67
Grupy nieprzechwytyjące	70
Czego dowiedziałeś się z rozdziału 4.?	73
Informacje techniczne	73
5. Klasy znaków	75
Negacja klasy znaków	77
Złączenia i różnice	77
Klasy znaków POSIX	80
Czego dowiedziałeś się z rozdziału 5.?	82
Informacje techniczne	82
6. Dopasowanie Unicode i innych znaków	83
Dopasowanie znaku Unicode	84
Dopasowanie znaków liczb ósemkowych	88
Dopasowanie właściwości znaku Unicode	88
Dopasowanie znaków kontrolnych	92
Czego dowiedziałeś się z rozdziału 6.?	94
Informacje techniczne	94

7. Kwantyfikatory	97
Zachłanne, leniwe i zaborcze	98
Dopasowanie za pomocą *, + oraz ?	99
Dopasowanie określoną liczbę razy	100
Kwantyfikatory leniwe	101
Kwantyfikatory zaborcze	103
Czego dowiedziałeś się z rozdziału 7.?	104
Informacje techniczne	104
8. Przewidywania	105
Przewidywanie pozytywne	105
Przewidywania negatywne	108
Pozytywne przewidywanie wsteczne	109
Negatywne przewidywanie wsteczne	109
Czego dowiedziałeś się z rozdziału 8.?	110
Informacje techniczne	110
9. Dodawanie znaczników HTML5 do dokumentu	111
Dopasowanie znaczników	112
Transformacja zwykłego tekstu za pomocą narzędzia sed	113
Dodawanie znaczników	117
Transformacja zwykłego tekstu za pomocą języka Perl	119
Czego dowiedziałeś się z rozdziału 9.?	125
Informacje techniczne	125
10. To już koniec	127
Dalsza nauka	129
Ważne narzędzia, implementacje i biblioteki	129
Dopasowanie numeru telefonu	132
w formacie stosowanym w Ameryce Północnej	132
Dopasowanie adresu e-mail	133
Czego dowiedziałeś się z rozdziału 10.?	134
A Odniesienia do wyrażeń regularnych	135
B Słownik wyrażeń regularnych	153
Skorowidz	163

Alternatywy, grupy i odniesienia

Miałeś już okazję zobaczyć grupy w działaniu. Tekst grupy znajduje się w nawiasie, a sama grupa ma za zadanie pomóc w wykonywaniu pewnych operacji, na przykład w:

- obsłudze alternatywy, czyli wyborze między co najmniej dwoma opcjonalnymi wzorcami;
- tworzeniu podwzorców;
- przechwytywaniu grupy w celu późniejszego odniesienia się do niej za pomocą odwołania wstecznego;
- zastosowaniu pewnej operacji do zgrupowanego wzorca, przykładowo kwantyfikatora;
- używaniu grup nieprzechwytyjących;
- używaniu grup niepodzielnych (temat zaawansowany).

Oprócz tekstu *The Rime of the Ancyent Mariner* z pliku *rime.txt* w tym rozdziale zostanie przedstawionych także kilka innych przykładów. Będę używać biurowej wersji aplikacji RegExr oraz innych narzędzi, na przykład sed. Biurową wersję aplikacji RegExr dla systemów Windows, Mac i Linux możesz pobrać z strony <http://www.gskinner.com/RegExr/desktop/> (aplikacja została utworzona w technologii Adobe AIR). Więcej informacji znajdziesz na wskazanej stronie.

Alternatywy

Ujmując rzecz najprościej, **alternatywa** pozwala na wybór wzorców do dopasowania. Przyjmujemy założenie, że chcesz się dowiedzieć, ile razy słowo *the* występuje w utworze *The Rime of the Ancyent Mariner*. Problem polega na tym, że to słowo jest w utworze pisane w postaci *THE*, *The* i *the*. Rozwiązaniem problemu jest wówczas użycie alternatywy.

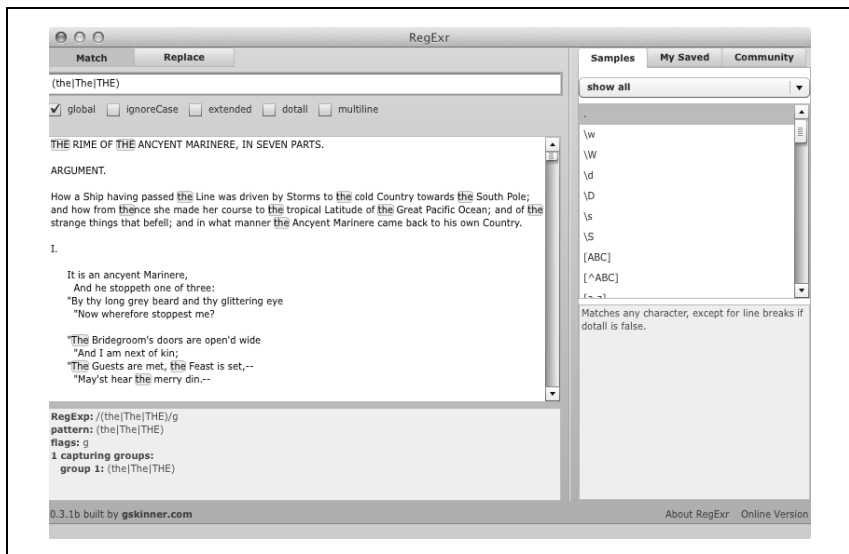
Uruchom biurową wersję aplikacji RegExr, dwukrotnie klikając jej ikonę. Aplikacja wygląda niemal tak samo jak wersja sieciowa, ale ma tę zaletę, że działa na komputerze lokalnym niezależnie od ewentualnych problemów związanych z siecią, które czasem pojawiają się podczas używania aplikacji sieciowych.

Na potrzeby omawianego ćwiczenia skopiowałem cały utwór z pliku *rime.txt*, a następnie wkleiłem go w aplikacji RegExr. Używana przeze mnie wersja systemu to Mac OS X 10.7 (Lion).

W górnym polu tekstowym aplikacji wprowadź wyrażenie regularne:

```
(the|The|THE)
```

które spowoduje podświetlenie w dolnym polu tekstowym wszystkich wystąpień słowa *the* w utworze (zobacz rysunek 4.1). Aby zobaczyć kolejne podświetlone wystąpienia słowa *the* w tekście docelowym, użyj przycisków przewijania.



Rysunek 4.1. Użycie alternatywy w biurowej wersji aplikacji RegExr

Istnieje możliwość skrócenia przedstawionej grupy poprzez zastosowanie opcji. Opcje pozwalają na określenie, jak ma być przeprowadzone wyszukiwanie wzorca. Przykładowo opcja:

(?i)

powoduje, że wzorzec nie rozróżnia wielkości liter. Dlatego też zamiast pierwotnego wzorca alternatywy możemy użyć poniższego:

(?i) the

Wypróbuj to wyrażenie regularne w aplikacji RegExr i przekonaj się, jaki będzie wynik jego działania. Ignorowanie wielkości liter można włączyć także poprzez zaznaczenie pola wyboru *ignoreCase* w aplikacji RegExr. Oba wymienione sposoby dają taki sam wynik. Przedstawioną opcję oraz kilka innych wymieniono w tabeli 4.1.

Tabela 4.1. Opcje wyrażen regularnych

Opcja	Opis	Obsługiwana przez
(?d)	Wiersze w systemie Unix	Java
(?i)	Wielkość liter	PCRE, Perl, Java
(?J)	Zezwolenie na duplikaty	PCRE ¹
(?m)	Multiline	PCRE, Perl, Java
(?s)	Pojedynczy wiersz (dotall)	PCRE, Perl, Java
(?u)	Wielkość liter w Unicode	Java
(?U)	Domyślne dopasowanie leniwe	PCRE
(?x)	Ignoruj znaki niewidoczne i komentarze	PCRE, Perl, Java
(?-...)	Włączenie lub wyłączenie opcji	PCRE

Teraz użyjemy alternatywy podczas pracy z narzędziem `grep`. Warto w tym miejscu wspomnieć, że opcje wymienione w tabeli 4.1 nie działają z narzędziem `grep`, więc konieczne będzie użycie pierwotnego wzorca alternatywy. Aby obliczyć liczbę wierszy, w których słowo *the* zapisane za pomocą znaków o dowolnej wielkości występuje przynajmniej jeden raz, wydaj następujące polecenie:

```
grep -Ec "(the|The|THE)" rime.txt
```

Otrzymane dane wyjściowe są proste:

```
327
```

¹ Zobacz sekcję „Named Subpatterns” na stronie <http://www.pcre.org/pcre.txt>.

Wynik nie mówi wszystkiego, ale o tym za chwilę.

Oto dokładna analiza wydanego polecenia narzędzia `grep`:

- opcja `-E` oznacza użycie rozszerzonych wyrażeń regularnych (ERE) zamiast zwykłych (BRE), dzięki czemu unikasz konieczności poprzedzania ukośnikami nawiasów i pionowych kresek, jak ma to miejsce w przypadku wyrażeń BRE, na przykład `\(THE\|The\|the\)`;
- opcja `-c` zwraca liczbę dopasowanych wierszy (a nie dopasowanych słów);
- w nawiasie znajduje się grupa alternatywy (*the*, *The* i *THE*);
- pionowa kreska oddziela poszczególne możliwości zapisu sprawdzane od lewej do prawej strony.

Poniższe polecenie zwróci wszystkie wystąpienia wskazanego słowa, wiersz po wierszu:

```
grep -Eo "(the|The|THE)" rime.txt | wc -l
```

Dane wyjściowe wykonanego polecenia to:

```
412
```

A oto jego dokładne omówienie:

- opcja `-o` oznacza wyświetlenie jedynie tej części wiersza, która została dopasowana do wzorca — w omawianym przykładzie dane nie będą jednak wyświetlone na ekranie, ponieważ są potokowane (`|`) do polecenia `wc`;
- w omawianym kontekście pionowa kreska powoduje przekazanie danych wyjściowych narzędzia `grep` jako danych wejściowych polecenia `wc` — polecenie `wc` oblicza liczbę słów, natomiast jego opcja `-l` zlicza liczbę wierszy danych wejściowych.

Skąd wzięła się tak duża różnica: jedno polecenie zwraca 327 wystąpień, natomiast drugie 412? Odpowiedź jest prosta: ponieważ opcja `-c` podaje liczbę dopasowanych wierszy, a przecież w wierszu może znajdować się więcej niż tylko jedno wystąpienie szukanego słowa. Jeżeli użyjemy opcji `-o` wraz z `wc -l`, wówczas każde wystąpienie słowa w dowolnej formie (wielkość znaków) zostanie umieszczone w oddzielnym wierszu i zliczone, co da w efekcie liczbę 412.

To samo zadanie, ale wykonane za pomocą języka Perl, wymaga użycia poniższego polecenia:

```
perl -ne 'print if /(the|The|THE)/' rime.txt
```

Jeszcze lepszym rozwiązaniem będzie użycie wspomnianej wcześniej opcji (?:i), ale bez alternatywy:

```
perl -ne 'print if /(?:i)the/' rime.txt
```

Jednak najlepsze rozwiązanie polega na dodaniu modyfikatora i po ostatnim ograniczniku wzorca:

```
perl -ne 'print if /the/i' rime.txt
```

Po wykonaniu powyższych poleceń otrzymasz takie same dane wyjściowe jak wcześniej. Im prostsze podejście, tym lepiej. Listę innych modyfikatorów (nazywanych także **flagami**) przedstawiono w tabeli 4.2. Porównaj też opcje (podobne, ale stosujące inną składnię) wymienione w tabeli 4.1.

Tabela 4.2. Modyfikatory (flagi) w języku Perl²

Modyfikator	Opis
a	Dopasowanie \d, \s, \w i POSIX jedynie w zakresie ASCII
c	Zachowaj bieżące położenie, jeśli próba dopasowania zakończy się niepowodzeniem
d	Użyj domyślnych, rodzimych reguł używanego systemu
g	Dopasowanie globalne
i	Dopasowanie bez uwzględnienia wielkości liter
l	Użycie reguł ustawień regionalnych użytkownika
m	Ciągi tekstowe obejmujące wiele wierszy
p	Zachowanie dopasowanego ciągu tekstowego
s	Traktowanie ciągów tekstowych jako pojedynczego wiersza
u	Użycie reguł Unicode podczas dopasowania
x	Ignorowanie znaków niewidocznych i komentarzy

Podwzorce

Bardzo często, odwołując się do **podwzorców** w wyrażeniach regularnych, odnosimy się do grupy lub grup wewnątrz innych grup. Podwzorzec jest więc wzorcem w innym wzorcu. Często zdarza się, że warunek zawarty w podwzorcu jest możliwy do spełnienia po dopasowaniu wcześniejszego wzorca, ale to nie jest regułą. Podwzorce można tworzyć na wiele różnych sposobów. W tym podrozdziale skoncentrujemy się przede wszystkim na podwzorcach definiowanych przez użycie nawiasów.

² Zobacz <http://perldoc.perl.org/perlre.html#Modifiers>.

W pewnym sensie użyty we wcześniejszej części rozdziału wzorzec:

(the|The|THE)

ma trzy podwzorce: pierwszy to *the*, drugi to *The*, a trzeci to *THE*. W omawianym przypadku dopasowanie na przykład drugiego podwzorca zupełnie nie zależy od dopasowania pierwszego. (Jako pierwszy zostanie dopasowany wzorzec pierwszy z lewej strony).

Poniżej widać przykład podwzorców, których dopasowanie zależy od dopasowania wcześniejszego wzorca:

(t|T)h(e|eir)

To wyrażenie regularne powoduje dopasowanie dosłownych znaków *t* lub *T*, następnie *h*, a dalej litery *e* lub liter *eir*. Dlatego też może dopasować dowolne z poniższych słów:

- *the*
- *The*
- *their*
- *Their*

W tym przypadku drugi podwzorzec (e|eir) jest zależny od pierwszego (t|T).

Podwzorce nie wymagają używania nawiasów. Poniżej przedstawiono definicję podwzorców, do której utworzenia wykorzystano klasy znaków:

`\b[tT]h[ceinry]*\b`

Powyższe wyrażenie regularne może dopasować oprócz słów *the* lub *The* także *thee*, *thy* i *thence*. Dwa wyrażenia granicy słowa (`\b`) powodują, że będzie ono dopasowywało całe słowa, a nie litery znajdujące się w innych słowach.

Oto dokładne omówienie przedstawionego wyrażenia regularnego:

- wyrażenie `\b` powoduje dopasowanie granicy początku słowa;
- `[tT]` to klasa znaków powodująca dopasowanie małej litery *t* lub dużej litery *T* — ten fragment wyrażenia regularnego możemy uznać za podwzorzec;
- następnie wzorzec dopasowuje (lub próbuje dopasować) małą literę *h*;
- drugi i zarazem ostatni podwzorzec również jest zdefiniowany w postaci klasy znaków `[ceinry]`, po której znajduje się kwantyfikator `*` określający dopasowanie zero lub więcej razy;
- na końcu wyrażenia regularnego mamy kolejne dopasowanie granicy słowa `\b`.



Jednym z bardziej interesujących aspektów wyrażeń regularnych jest to, że stosowana terminologia może być blisko związana ze znaczeniem danej operacji lub wręcz przeciwnie. Decydując się na użycie terminu *podwzorzec* oraz innych w niniejszej książce, przeanalizowałem wiele źródeł i spróbowałem je wszystkie połączyć. Na pewno znajdą się Czytelnicy, którzy będą się upierać, że klasa znaków nie jest podwzorcem. Uważam, że skoro klasy znaków mogą działać tak jak podwzorze, to mogą je wrzucić do jednego worka.

Grupy przechwytywania i odwołania wsteczne

Kiedy wzorzec grupuje całą treść lub jej część w nawiasach, wówczas przechwytuje tę treść i przechowuje ją tymczasowo w pamięci. Następnie, jeśli chcesz, możesz ponownie wykorzystać tę treść, używając odwołania wstecznego w postaci:

`\1`

lub:

`$1`

przy czym `\1` lub `$1` odwołują się do pierwszej grupy przechwytywania, `\2` lub `$2` — do drugiej itd. Narzędzie `sed` akceptuje jedynie postać `\1`, natomiast język Perl dopuszcza obie.



Początkowo narzędzie `sed` obsługiwało odwołania wsteczne w zakresie od `\1` do `\9`, ale takie ograniczenie już nie istnieje.

Odwołania wsteczne już widziałeś w działaniu, ale przedstawię je raz jeszcze. W omawianym przykładzie wykorzystamy je do zmiany kolejności słów w wierszu utworu, za co przeproszam Samuela Taylora Coleridge’a. W aplikacji RegExr kliknij kartę *Replace*, a następnie w górnym polu tekstowym wprowadź wzorzec:

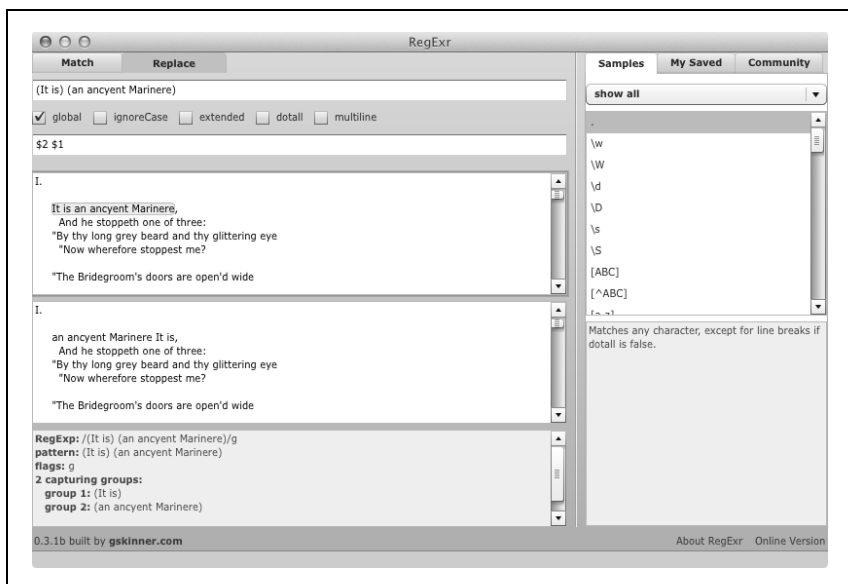
`(It is) (an ancyent Marinere)`

Przewijaj trzecie pole tekstowe zawierające tekst docelowy, aż zobaczysz podświetlony wiersz, a później w drugim polu tekstowym wprowadź takie wyrażenie:

`$2 $1`

W dolnym polu tekstowym zobaczysz, że słowa w podświetlonym wierszu zostały zamienione miejscami (spójrz na rysunek 4.2).

an ancylent Marinere It is,



Rysunek 4.2. Odwołania wsteczne za pomocą wyrażeń \$1 i \$2

Poniższe polecenie pozwala na wykonanie tego samego zadania, ale za pomocą narzędzia sed:

```
sed -En 's/(It is) (an ancylent Marinere)/\2 \1/p' rime.txt
```

Otrzymane dane wyjściowe polecenia są następujące:

an ancylent Marinere It is,

Wynik operacji jest więc dokładnie taki sam, jak uzyskany w aplikacji RegExr. Poniżej przedstawiono dokładne omówienie działania wywołania narzędzia sed. Dzięki temu możesz dobrze zrozumieć, jak naprawdę działa omawiane polecenie.

- opcja -E powoduje użycie rozszerzonych wyrażeń regularnych (ERE), co zwalnia Cię na przykład z konieczności poprzedzania nawiasów ukośnikami;
- opcja -n zawiesza zachowanie domyślne polegające na wyświetleniu każdego wiersza;

- polecenie zastępowania powoduje wyszukanie dopasowanego tekstu *It is an ancylent Marinere* i przechwytyje go w dwóch grupach;
- polecenie zastępowania powoduje także zmianę kolejności słów w dopasowanym tekście, używając do tego odwołań wstecznych: najpierw \2, a później \1;
- opcja p na końcu polecenia zastępowania oznacza wyświetlenie zmodyfikowanego wiersza.

Polecenie w języku Perl wykonujące takie samo zadanie ma postać:

```
perl -ne 'print if s/(It is) (an ancylent Marinere)/\2 \1/' rime.txt
```

Zwróć uwagę na użycie składni w stylu \1. Oczywiście masz również możliwość użycia składni \$1:

```
perl -ne 'print if s/(It is) (an ancylent Marinere)/$2 $1/' rime.txt
```

Bardzo podoba mi się sposób, w jaki Perl pozwala na wyświetlenie wskazanego wiersza. Chciałbym jeszcze zwrócić uwagę na jedną rzecz w wyświetlonych danych wyjściowych:

```
an ancylent Marinere It is,
```

Podczas transformacji została zachowana wielkość liter. Perl pozwala rozwiązać ten problem dzięki użyciu \u i \l:

```
perl -ne 'print if s/(It is) (an ancylent Marinere)/\u$2 \l$1/' rime.txt
```

Teraz otrzymane dane wyjściowe przedstawiają się znacznie lepiej:

```
An ancylent Marinere it is,
```

Oto dokładne omówienie działania wyrażeń \u i \l:

- składnia \l nie dopasowuje żadnego znaku, ale zmienia znajdujący się po niej znak na mały;
- składnia \u zmienia znajdujący się po niej znak na wielki;
- dyrektywa \U (nieużyta w przykładzie) zmienia cały ciąg tekstowy znajdujący się po niej na zapisany wielkimi literami;
- dyrektywa \L (nieużyta w przykładzie) zmienia cały ciąg tekstowy znajdujący się po niej na zapisany małymi literami.

Wymienione dyrektywy działają aż do wystąpienia kolejnej (w przypadku \l lub \E do końca ciągu tekstowego). Poeksperymentuj z nimi i przekonaj się, jaki jest ich sposób działania.

Nazwane grupy

Nazwane grupy to po prostu grupy przechwytywania wraz ze zdefiniowanymi nazwami. Dostęp do takich grup odbywa się poprzez ich nazwy, a nie liczby. Oto sposób użycia nazwanych grup w języku Perl:

```
perl -ne 'print if s/(?<one>It is) (?<two>an ancyent Marinere)/\u${two}
\l${one}/' rime.txt
```

Powyższe polecenie działa następująco:

- dodanie `?<one>` i `?<two>` w nawiasach powoduje zdefiniowanie nazw dla danych grup, odpowiednio *one* i *two*;
- `${one}` odwołuje się do grupy o nazwie *one*, natomiast `${two}` — do grupy o nazwie *two*.

Nazwanych grup można również używać wewnątrz wzorca, w którym zostały nazwane. Poniżej wyjaśnię Ci, co to oznacza. Przyjmujemy założenie, że szukany jest ciąg tekstowy składający się z sześciu zer:

```
000000
```

To naciągany przykład, ale ma za zadanie jedynie zademonstrować sposób działania. Grupę trzech zer w podanym wzorcu nazywamy *z* (można użyć dowolnej nazwy):

```
(?<z>0{3})
```

Następnie tak nazwanej grupy można użyć ponownie w taki sposób:

```
(?<z>0{3})\k<z>
```

taki:

```
(?<z>0{3})\k'z'
```

lub taki:

```
(?<z>0{3})\g{z}
```

Wypróbuj powyższe wyrażenia regularne w aplikacji RegExr i przekonaj się, jaki jest wynik ich działania. W tabeli 4.3 wymieniono wiele przykładów składni odwoływania się do nazwanych grup.

Grupy nieprzechwytyjące

Istnieją również tak zwane grupy nieprzechwytyjące, to znaczy takie, które nie przechowują swojej zawartości w pamięci. Czasami takie rozwiązanie będzie zaletą, zwłaszcza jeśli nigdy nie masz zamiaru odwołać się do grupy.

Tabela 4.3. Składnia nazwanych grup

Składnia	Opis
<code>(?<nazwa>...)</code>	Nazwana grupa
<code>(?nazwa...)</code>	Inna nazwana grupa
<code>(?P<nazwa>...)</code>	Nazwana grupa (w języku Python)
<code>\k<nazwa></code>	Odniesienie poprzez nazwę (w języku Perl)
<code>\k'nazwa'</code>	Odniesienie poprzez nazwę (w języku Perl)
<code>\g{nazwa}</code>	Odniesienie poprzez nazwę (w języku Perl)
<code>\k{nazwa}</code>	Odniesienie poprzez nazwę (w językach .NET)
<code>(?P=nazwa)</code>	Odniesienie poprzez nazwę (w języku Python)

Ponieważ grupa nie przechowuje swojej treści, jej działanie może charakteryzować się większą wydajnością. Jednak w przypadku tak prostych przykładów jak prezentowane w niniejszej książce kwestie związane z wydajnością nie występują.

Pamiętasz pierwszą grupę przedstawioną w tym rozdziale? Dla przypomnienia:

```
(the|The|THE)
```

Ponieważ nie ma potrzeby odwoływania się do powyższej grupy, można ją zdefiniować jako grupę nieprzechwytyjącą. Wymaga to wyrażenia regularnego w następującej postaci:

```
(?:the|The|THE)
```

Opierając się na materiale przedstawionym na początku rozdziału, powyższe wyrażenie można uzupełnić o opcję powodującą, że wzorzec nie rozróżnia wielkości liter (choć w przypadku tej grupy nie trzeba dodawać wymienionej opcji):

```
(?i)(?:the)
```

Inny sposób:

```
(?:(?i)the)
```

Najlepiej jednak użyć poniższego zapisu:

```
(?i:the)
```

Litera `i` i opcji może zostać umieszczona pomiędzy znakiem zapytania i dwukropkiem.

Grupy niepodzielne

Inny rodzaj grupy nieprzechwytyjącej to tak zwana **grupa niepodzielna**. Jeżeli korzystasz z silnika wyrażeń regularnych obsługującego sprawdzanie wsteczne (ang. *backtracking*), taka grupa spowoduje wyłączenie sprawdzania wstecznego, ale nie w przypadku silnika wyrażeń regularnych, lecz tylko w odniesieniu do części wyrażenia ujętej w grupie niepodzielnej. Składnia przedstawia się tak:

(?>the)

Do czego mogą służyć grupy niepodzielne? Jednym z czynników, który może naprawdę spowolnić przetwarzanie wyrażenia regularnego, jest właśnie sprawdzanie wsteczne. Powód jest prosty: sprawdzenie wszystkich możliwości wymaga czasu i zasobów komputera. Czasami przetworzenie wyrażenia regularnego może pochłonąć naprawdę ogromną ilość czasu. Kiedy sytuacja staje się poważna, używane jest określenie **katastrofalne sprawdzanie wsteczne**.

Sprawdzanie wsteczne można wyłączyć, używając silnika wyrażeń regularnych pozbawionego jego obsługi, na przykład `re2` (<http://code.google.com/p/re2/>), lub wyłączając sprawdzanie wsteczne dla fragmentów wyrażenia regularnego przez użycie grup niepodzielnych.



W niniejszej książce skoncentrowałem się na przedstawieniu składni, dlatego niewiele miejsca poświęcę na omówienie zagadnień optymalizacji prowadzącej do zwiększenia wydajności działania wyrażeń regularnych. Z mojego punktu widzenia grupy podzielne są stosowane przede wszystkim ze względu na wydajność.

Z rozdziału 5. dowiesz się więcej na temat klas znaków.

Czego dowiedziałeś się z rozdziału 4.?

- Alternatywa pozwala na wybranie z dwóch lub większej liczby wzorców.
- Czym są modyfikatory oraz jak można ich używać we wzorcach?
- Jakie są rodzaje podwzorców?
- Jak używać grup przechwytywania oraz odwołań wstecznych?
- Jak używać nazwanych grup oraz jak odwoływać się do nich?
- Jak używać grup nieprzechwytyjących?
- Co to są grupy niepodzielne?

Informacje techniczne

- Środowisko Adobe AIR pozwala na skorzystanie z technologii HTML, JavaScript, Flash i ActionScript do tworzenia aplikacji sieciowych działających jako samodzielne aplikacje po stronie klienta bez konieczności używania przeglądarki internetowej do ich uruchamiania. Więcej informacji na temat Adobe AIR znajdziesz na stronie <http://www.adobe.com/pl/products/air.html>.
- Python (<http://www.python.org/>) to łatwy do zrozumienia język programowania wysokiego poziomu. Python zawiera implementację wyrażeń regularnych (zobacz <http://docs.python.org/2/library/re.html>).
- .NET (<http://www.microsoft.com/net>) to platforma programistyczna opracowana dla Windowsa. Ona również zawiera implementację wyrażeń regularnych (zobacz <http://msdn.microsoft.com/en-us/library/hs600312.aspx>).
- Bardziej zaawansowane objaśnienie tematu grup niepodzielnych znajdziesz na stronach <http://www.regular-expressions.info/atomic.html> i <http://stackoverflow.com/questions/6488944/atomic-group-and-non-capturing-group>.

\$1, 67	[[[:word:]], 81
(?-...), 63	[[[:xdigit:]], 81
(?d), 63	[0-9], 17
(?i), 63	[\b], 34
(?j), 63	\0, 34, 92
(?m), 63	\1, 19, 67
(?s), 63	\a, 34
(?u), 63	\A, 52
(?U), 63	\cx, 34, 92
(?x), 63	\d, 13, 18, 34
*, znak, 20	\D, 18, 32, 34
.*, 99	\dxx, 34
., znak, 19	\E, 53
.NET, 73	\f, 36
?, znak, 20	\h, 36
[[[:^alpha:]], 81	\H, 36
[[[:^xxx:]], 81	\l, 69
[[[:alnum:]], 81	\L, 69
[[[:alpha:]], 81	\n, 35, 36
[[[:ascii:]], 81	\oxxx, 34
[[[:blank:]], 81	\Q, 53
[[[:ctrl:]], 81	\r, 35, 36
[[[:digit:]], 81	\s, 35, 36
[[[:graph:]], 81	\S, 36
[[[:lower:]], 81	\t, 35, 36
[[[:print:]], 81	\u, 69
[[[:punct:]], 81	\U, 69
[[[:space:]], 81	\uxxxx, 34
[[[:upper:]], 81	\v, 36

\V, 36
\w, 32, 34
\W, 33, 34
\x xx, 34
\z, 52
\Z, 52
+, znak, 20

A

ack, narzędzie, 88, 89, 95
Adobe AIR, 73
alarm, 34
alternatywy, 62, 153
ASCII, 83, 153
 tablica kodów, 147, 148, 149, 150, 151
AsciiDoc, 125
asercja, 47, 153
 o zerowej długości, 47, 105, 153
atomy, *Patrz* metaznaki

B

Backspace, dopasowanie, 34, 93
Bell, dopasowanie, 92
BRE, 51, 153, 157
bufor roboczy, *Patrz* przestrzeń wzorca

C

code point, *Patrz* punkt kodowy
composability, 26, 160
cyfry, dopasowanie, 17, 18, 30, 34
Cygwin, 10

D

dopasowanie, 154
 alarm, 34
 Backspace, znak, 34, 93
 Bell, znak, 92
 cyfry, 18, 34

e-mail, 133
Esc, znak, 93
inny niż cyfra, znak, 18, 32, 34
inny niż niewidoczny, znak, 36
inny niż spacja, znak, 36
kontrolny, znak, 34, 92
leniwe, 154
liczba ósemkowa, 88
niepionowy tabulator, 36
niepoziomy znak niewidoczny, 36
niesłowo, 33, 34
niewidoczny, znak, 35
nowy wiersz, znak, 35, 36
null, znak, 34, 92
numer telefonu, 132
określoną liczbę razy, 100
pionowy tabulator, 36
początek i koniec wiersza, 47, 48
powrót na początek wiersza,
 znak, 35, 36

poziomy tabulator, 36
poziomy znak niewidoczny, 36
słowo, 32, 34
spacja, 36
tabulator, 35
Unicode, 34, 84
wartość dziesiętna znaku, 34
wartość ósemkowa znaku, 34
wartość szesnastkowa znaku, 34
wysunięcie strony, znak, 36
zaborcze, 154
zachłanne, 154
znaczniki, 112

dosłowny ciąg tekstowy, 17, 29, 154

E

echo, polecenie, 40
ed, edytor, 154
egrep, narzędzie, 155
e-mail, dopasowanie, 133
ERE, 51, 154, 158
Esc, dopasowanie, 93

F

fgrep, narzędzie, 155
flaga, *Patrz* modyfikator
fragment, 154

G

gałąź, 155
Git, 44
GitHub, 44
granice, *Patrz* kwantyfikatory
grep, narzędzie, 51, 59, 155
grupy, 155
 niepodzielne, 72, 155
 nieprzechwytyjące, 70, 155
 przechwytywania, 19, 67, 155
gwiazda Kleene’a, 99
gwiazdka, znak, 20

H

HTML5, 44

K

katastrofalne sprawdzanie wsteczne,
 72, 156
klasy znaków, 17, 75, 76, 77, 156
 negacja, 77
 POSIX, 80, 81, 145
Kleene, Stephen, 7
kotwice, 47, 52, 156
kropka, znak, 19, 37
kwantyfikatory, 20, 97, 98, 156
 leniwe, 98, 101, 102
 podstawowe, 100
 zaborcze, 98, 103, 104
 zachłanne, 98

L

Levithan, Steven, 27
literał, 29, 156
Lovitt, Michael, 82

M

McMahon, Lee, 40
metaznaki, 17, 137, 156
modyfikatory, 146, 156

N

nazwane grupy, 70, 71
negacja, 157
negatywne przewidywanie wsteczne,
 109, 157
niepionowy tabulator, dopasowanie,
 36
niepoziomy znak niewidoczny,
 dopasowanie, 36
niesłowo
 dopasowanie, 33, 34
 granice, 49, 50
Notepad++, edytor, 24, 26
null, dopasowanie, 34, 92
numer telefonu, dopasowanie, 16, 17,
 18, 19, 20, 21, 22, 132

O

odwołania wsteczne, 19, 67, 157
ograniczone wystąpienia, *Patrz*
 kwantyfikatory
Oniguruma, 131
opcje, 146, 157
Opera Next, 82
Oxygen, edytor, 24, 26

P

PCRE, 59, 130
pcgrep, 52, 130
Perl, 42, 43, 45, 130, 157
 dodawanie znaczników, 57, 119
 modyfikatory, 65, 146
 obsługa liczb rzymskich, 122
 obsługa wierszy, 122
 plik poleceń, 123
perldoc, 126
pionowy tabulator, dopasowanie, 36
plus, znak, 20
podstawowe wyrażenia regularne,
 Patrz BRE
podwzorzec, 65, 66
POSIX, 80, 82, 157
 klasy znaków, 145
poziomy tabulator, dopasowanie, 36
poziomy znak niewidoczny,
 dopasowanie, 36
pozytywne przewidywanie wsteczne,
 109, 157
przestrzeń wzorca, 158
przewidywania, 105, 158
 negatywne, 108
 pozytywne, 105
 wstecz, 158
punkt kodowy, 14, 158
Python, 73, 131

Q

QED, edytor, 7, 13, 135
 wyrażenia regularne, 136

R

Rackham, Stuart, 125
RE2, 132
Regex Hero, aplikacja, 85, 94
RegexPal, 14, 15, 26

RegExr, 27, 28, 44
 dodawanie znaczników, 54, 55
Reggy, aplikacja, 78, 82
rozszerzone wyrażenia regularne,
 Patrz ERE
Rubular, aplikacja, 80, 82
Ruby, 131

S

sed, narzędzie, 40, 44, 159
 dodawanie znaczników, 55, 56,
 113
 obsługa liczb rzymskich, 115
 obsługa wierszy, 116
 plik poleceń, 118
 zastępowanie znaków, 114
Skinner, Grant, 27
słowo
 dopasowanie, 32, 34
 granice, 49, 50
spacja, dopasowanie, 36
sprawdzanie wsteczne, 98, 159
system szesnastkowy, 159

T

tabulator, dopasowanie, 35
TextMate, edytor, 23, 26
Thomson, Ken, 7, 13, 59, 154, 155

U

Unicode, 83, 159
 dla znaków niewidocznych, 139,
 140
 dopasowanie, 34, 84
 właściwości znaku, 88, 90, 91
utrzymywana przestrzeń, 160
utrzymywany bufor, *Patrz*
 utrzymywana przestrzeń

V

vi, edytor, 51, 59, 160
vim, edytor, 51, 59, 94, 160
dopasowanie znaku Unicode, 87

W

Wall, Larry, 42
wartość dziesiętna znaku,
dopasowanie, 34
wartość ósemkowa znaku,
dopasowanie, 34
wartość szesnastkowa znaku,
dopasowanie, 34
wiersz, początek i koniec, 47, 48
właściwości znaku, 142, 143
wyrażenia regularne, 7, 13, 161
opcje, 63
wyrażenia w nawiasach,
Patrz klasy znaków
wzorce, 27

Z

zakres, 31
składnia, 101
zestaw znaków, 17, 77, 161
znaczniki, dopasowanie, 112

znak inny niż cyfra, dopasowanie, 18,
32, 34
znak inny niż spacja, dopasowanie,
36
znak kontrolny, dopasowanie, 34, 92,
140, 141
znak niewidoczny, dopasowanie, 35,
139, 140
znak nowego wiersza, dopasowanie,
35, 36
znak ósemkowy, 161
znak powrotu na początek wiersza,
dopasowanie, 35, 36
znak sterujący, 18, 161
znak wysunięcia strony,
dopasowanie, 36
znak zapytania, 20
znaki skrótów, 14, 18, 138
dla znaków niewidocznych, 36,
139
lista, 34

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA



Helion SA

Wyrażenia regularne. Wprowadzenie



Wyrażenia regularne to wzorce, które pozwalają opisać łańcuchy znaków. Brzmi to groźnie, wygląda jeszcze gorzej, ale każdy programista prędzej czy później się z nimi spotka i... doceni ich potęgę! Trudno sobie wyobrazić wyszukiwanie, zastępowanie oraz sprawdzanie poprawności danych bez wykorzystania potencjału wyrażeń regularnych. Czas poświęcony na ich opanowanie zwróci się błyskawicznie i z nawiązką!

Ta wspaniała książka wprowadzi Cię w świat wyrażeń regularnych szybko i bezboleśnie. Już za chwilę wykorzystasz podstawowe elementy wyrażeń, a każdy kolejny rozdział dostarczy Ci coraz bardziej zaawansowanych narzędzi. W trakcie lektury nauczysz się korzystać z granic, klas znaków, grup i odniesień. Ponadto dowiesz się, jak wykorzystać możliwości Perla w zakresie transformacji tekstów. Ten wyjątkowy podręcznik musi się znaleźć na półce każdego programisty!

Sprawdź już teraz:

- potencjał, jaki kryją wyrażenia regularne
- metody szybkiego wyszukiwania i zastępowania ciągów znaków
- sposoby korzystania z wyrażeń w różnych językach programowania
- możliwości wyrażeń regularnych w zakresie kontroli poprawności wprowadzonych danych

Zaoszczędź czas dzięki wyrażeniom regularnym!

helion.pl
księgarnia
internetowa

Nr katalogowy: 13618



Księgarnia internetowa:
<http://helion.pl>



Zamówienia telefoniczne:
0 801 339900
0 601 339900



Helion

Sprawdź najnowsze promocje:

➤ <http://helion.pl/promocje>

Książki najchętniej czytane:

➤ <http://helion.pl/bestsellery>

Zamów informacje o nowościach:

➤ <http://helion.pl/nowosci>

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel.: 32 230 98 63

e-mail: helion@helion.pl

<http://helion.pl>

sięgnij po WIĘCEJ



KOD KORZYŚCI

ISBN 978-83-246-6868-7



Cena 34,90 zł