Wyrażenia regularne

Regular expressions

aka

Regexp

Wyrażenia regularne

Historia:

- *teoria automatów oraz teoria języków formalnych*
- *Stephen Cole Kleene (1909-1994) – lata 50 – badania nad zbiorami regularnymi oraz teorią rekursji*
- *język programowania SNOBOL (1962-1968) Bell Labs – koncepcja dopasowywania wzorców*
- *edytor QED (1965-1966) Bell Labs – Ken Thompson – zastosowanie wyrażeń Kleene'a*

Wyrażenia regularne

Historia:

- *po przychylnym odbiorze swojego pomysłu Thompson wbudował regexpy również w edytor ed (a tym samym w vi)*
- *narzędzie grep – jego nazwa wzięła się od polecenia edytora ed o postaci:*

```
g/re/p
```

gdzie „re” oznacza właśnie regexp

Wyrażenia regularne

Historia:

- *kolejne narzędzia wykorzystujące regexp:*
  - expr
  - AWK *(Aho-Weinberger-Kernighan)*
  - Emacs
  - lex

Wyrażenia regularne

Historia:

- *języki programowania, które wbudowały regexp na poziomie składni:*
  - *Perl (1987 – Larry Wall - NASA)*
  - *Tcl (1988 - John Ousterhout – University of California, Berkeley)*

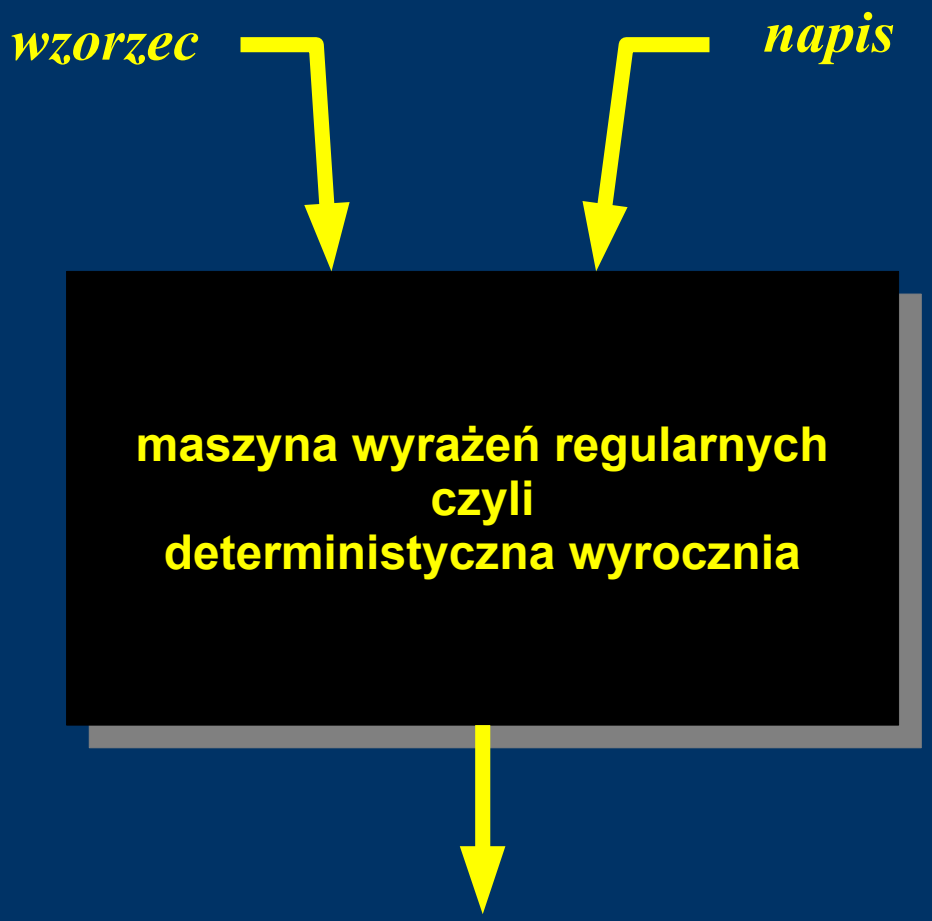
Wyrażenia regularne

Historia:

- *Perl Compatible Regular Expressions (PCRE) – Philip Hazel – implementacja regexp w stylu Perla w postaci biblioteki dla języka „C” - de facto standard*
- *adaptacja mechanizmów regexp dla większości współczesnych języków programowania: Java, JavaScript, Python, Ruby, rodzina .NET, XML Schema, PHP, różne implemenatcje SQL, etc.*

Wyrażenia regularne

Koncepcja:



Wyrażenia regularne

Zasada:

maszyna dopasowuje wzorzec do napisu analizując oba ciągi znaków od lewej do prawej.

Wyrażenia regularne

Zasada:

jeśli pewien znak wzorca nie jest znakiem zastrzeżonym, to dopasowuje się sam do siebie; w szczególności same do siebie dopasowują się litery i cyfry

Wyrażenia regularne

Przykład:

Napis: `"Ala ma kota"`

wzorzec	*dopasowanie*
`"Ala"`	TAK
`"la"`	TAK
`"mak"`	NIE
`"kota"`	TAK
`"ota"`	TAK
`"koty"`	NIE

Wyrażenia regularne

Zasada:

niektóre znaki wzorca nie dopasowują się same do siebie - są to tzw. ***metaznaki****.*

metaznaki języka wzorca wyrażeń regularnych:

```
\ | ( ) [ { ^ $ * + ? .
```

Wyrażenia regularne

Zasada:

jeśli \ poprzedza liczbę i liczba ta nie rozpoczyna się cyfrą 0, to jest to tzw. odwołanie wsteczne (backreference)

Przykład:

`\1` - *odwołanie wsteczne do pierwszego dopasowanego podłańcucha*

Wyrażenia regularne

Zasada:

jeśli `\0` ***poprzedza trzy cyfry ósemkowe, to dopasowuje się do znaku w napisie o kodzie wyrażonym zapisem ósemkowym (tzw. octal escape)***

Przykład:

`\040` – *dopasowuje się do spacji*

Wyrażenia regularne

Zasada:

jeśli `\x` poprzedza dwie cyfry szesnastkowe, to dopasowuje się do znaku w napisie o kodzie wyrażonym zapisem szesnastkowym (tzw. hex escape)

Przykład:

`\x20` – *dopasowuje się do spacji*

Wyrażenia regularne

Zasada:

jeśli `\c` *poprzedza dowolny znak alfabetyczny (*`A-Z` *lub* `a-z`*), to dopasowuje się do odpowiadającego mu znaku kontrolnego w napisie.*

Przykład:

`\cd` – *dopasowuje się do znaku* `^D`

Wyrażenia regularne

Zasada:

jeśli `\u` poprzedza cztery cyfry szesnatkowe, to dopasowuje się do znaku w napisie o kodzie wyrażonym zapisem szesnastkowym w standardzie UNICODE (tzw. unicode escape).

Przykład:

`\u00A9` – *dopasowuje się do znaku ©*

Wyrażenia regularne

Zasada:

jeśli \ poprzedza jedną z wymienionych dalej liter, to dopasowuje się do znaków specjalnych lub do jednego z klasy znaków:

Wyrażenia regularne

Znaki specjalne

\a	BEL
\n	LF
\r	CR
\t	HT
\f	FF
\e	ESC
\d	<dowolna cyfra dziesiętna>
\D	<znak inny niż cyfra dziesiętna>
\w	<cyfra, litera, podkreślnik>
\W	<nie cyfra, nie litera i nie podkreślnik>
\s	<biały znak spacja lub **\t** lub **\n** lub **\r** lub **\f**
\S	<nie-biały znak>

Wyrażenia regularne

Zasada:

***jeśli \ poprzedza dowolny inny znak (z kilkoma wyjątkami, o których niebawem) to jest to cytowanie (escaping); cytowania wymagają m.in. metaznaki oraz sam znak *** (*)

(*) *pamiętać należy o tym, że sam znak \ może wymagać osobnego cytowania zgodnie z wymogami języka C i pochodnych*

Wyrażenia regularne

Przykład:

napis	wzorzec	?
"Ala ma 12 kotów"	"\d\d"	TAK
"Ala ma 12 kotów"	"\d\d\d"	NIE
"Ola ma psa"	"\w\w\w"	TAK
"Ola ma psa"	"\w\w\w\w"	NIE
"Ola.ma.psa"	"\.\w\w\."	TAK

Wyrażenia regularne

Zasada:

metaznak ^ (caret) dopasowuje się nie do znaku, a do miejsca w napisie, tzn:

- *do początku napisu (w trybie single-line)*
- *do początku linii (w trybie multi-line)*

Wyrażenia regularne

Przykład: *napis:* "`Ala ma 3 koty`"

wzorzec	?
"Ala"	TAK
"^Ala"	TAK
"ma"	TAK
"^ma"	NIE
"^\D"	TAK
"\d"	TAK
"^\d"	NIE

Wyrażenia regularne

Zasada:

metaznak `$` (dollar) dopasowuje się nie do znaku, a do miejsca w napisie, tzn:

- *do końca napisu (w trybie single-line)*
- *do końca linii (w trybie multi-line)*

Wyrażenia regularne

Przykład: *napis:* "**Ala ma 3 koty**"

wzorzec	*?*
"Ala"	TAK
"Ala$"	NIE
"koty"	TAK
"koty$"	TAK
"\D$"	TAK
"\d$"	NIE
" \w\w\w\w$"	TAK

Wyrażenia regularne

Zasada:

- *metaznak* **`\b`** *dopasowuje się tylko na granicy słowa*
- *metaznak* **`\B`** *dopasowuje się wszędzie poza granicą słowa.*

Wyrażenia regularne

Przykład: *napis:* "wielka **rzeka** na**rzeka**"

wzorzec	*?*
"rzeka"	TAK
"\brzeka"	**TAK**
"\Brzeka"	**TAK**

Wyrażenia regularne

Zasada:

- *metaznak* **`\A`** *dopasowuje się do początku napisu*
- *metaznak* **`\Z`** *dopasowuje się do końca napisu*

Oba dopasowania występują w napisie raz, niezależnie od tego, czy maszyna jest w trybie single-line czy multi-line

Wyrażenia regularne

Zasada:

metaznak **{** *to tzw. kwantyfikator określający wymagania ilościowe odnoszące się do poprzedzającego go znaku/wyrażenia wzorca.*

Wyrażenia regularne

Kwantyfikatory:

zachłanny	*niezachłanny*	*wymagany zakres*
`{n,m}`	`{n,m}?`	musi się pojawić co najmniej *n* razy ale nie więcej niż *m* razy
`{n,}`	`{n,}?`	musi się pojawić co najmniej *n* razy
`{n}`	`{n}?`	musi się pojawić dokładnie *n* razy

dopasowywanie wzorca jest ZACHŁANNE jeśli nie użyto znaku ? i NIEZACHŁANNE, jeśli użyto znaku ?

Wyrażenia regularne

Przykład: *napis:* ″123123123123″

wzorzec	*dopasowanie*
″\d{0,12}123″	123123123123
″\d{0,12}?123″	123
″\d{3,9}123″	123123123123
″\d{3,9}?123″	123123
″\d{3,}123″	123123123123
″\d{3,}?123″	123123
″\d{3}123″	123123
″\d{3}?123″	123123

Wyrażenia regularne

Kwantyfikatory specjalne:

zachłanny	*niezachłanny*	*równoważnik*
`*`	`*?`	0 lub więcej razy `{0,}`
`+`	`+?`	1 lub więcej razy `{1,}`
`?`	`??`	0 lub 1 raz `{0,1}`

Wyrażenia regularne

Przykład: *napis:* "abcabcabc"

wzorzec	*dopasowanie*
"\D*abc"	abcabcabc
"\D*?abc"	abc
"\D+abc"	abcabcabc
"\D+?abc"	abcabc

Wyrażenia regularne

Zasada:

ujęcie fragmentu wyrażenia regularnego w parę nawiasów **( )** *nie zmienia jego znaczenia, a powoduje powstanie tzw. odwołania wstecznego (backreference); odwołań wstecznych może być dowolna liczba.*

Wyrażenia regularne

Przykład: *napis:* "1221"

wzorzec	*dopasowanie?*
"(\d)(\d)\2\1"	TAK
"(\d)(\d)\1\2"	NIE

Wyrażenia regularne

Zasada:

metaznak **.** *(kropka) pasuje do każdego znaku poza* **\n** *(w zasadzie)*

Wyrażenia regularne

Zasada:

lista znaków w nawiasach `[]` *(zbiór) dopasowuje jeden z tych znaków*

Wyrażenia regularne

Przykład: *wzorzec:* "`[OA]la`"

napis	*dopasowanie*
`Ola ma psa`	`TAK`
`Ala ma kota`	`TAK`

Wyrażenia regularne

Zasada:

znak **^** *użyty jako pierwszy znak w zbiorze powoduje, że dopasowanie będzie zachodziło tylko dla znaków, których NIE MA zbiorze (dopełnienie)*

Wyrażenia regularne

Przykład: *wzorzec:* "`[^OA]la`"

napis	*dopasowanie*
Ola ma psa	NIE
Ala ma kota	NIE
Dla psa kiełbasa	TAK

Wyrażenia regularne

Zasada:

wewnątrz zbioru dopuszcza się użycie znaku **-** *dla oznaczenia klas znaków, np.*

`[A-Z]`	*- wielkie litery*
`[a-z]`	*- małe litery*
`[0-9]`	*- cyfry*
`[A-Fa-f0-9]`	*- cyfry szesnastkowe*

Wyrażenia regularne

Zasada:

metaznak **|** *(bar) jest operatorem alternatywy*

Wyrażenia regularne

Przykład: *wzorzec:* "`(O|A)la`"

napis	*dopasowanie*
`Ola ma psa`	`TAK`
`Ala ma kota`	`TAK`
`Dla psa kiełbasa`	`NIE`

Wyrażenia regularne

Przykład: *wzorzec:* "`O|Ala`"

napis	*dopasowanie*
`Ola ma psa`	`NIE`
`Ala ma kota`	`TAK`
`O! To pies Ali!`	`TAK`

Wyrażenia regularne

Zasada:

ujęcie fragmentu wzorca w nawiasy `(?` *oraz* `)` *nie powoduje powstania odwołania wstecznego, a jedynie wyróżnienie podłańcucha (np. w celu użycia metaznaku* `|`*)*

Wyrażenia regularne

Przykład: *wzorzec:* `"(?O|A)la"`

napis	dopasowanie
Ola ma psa	TAK
Ala ma kota	TAK
Dla psa kiełbasa	NIE

Wyrażenia regularne

Zasada: pozytywne dopasowanie w przód

ujęcie fragmentu wzorca w nawiasy `(?=` *oraz* `)` *nie powoduje powstania odwołania wstecznego, a powoduje, że poszukiwanie następnego dopasowanie rozpocznie się przed dopasowanym właśnie fragmentem*

Wyrażenia regularne

Przykład: ***wymaganie: hasło ma mieć od 4 do 8 znaków i musi zawierać co najmniej jedną cyfrę***

Wzorzec:

```
"^(?=.*\d).{4,8}$"
```

Wyrażenia regularne

Zasada: negatywne dopasowanie w przód

ujęcie fragmentu wzorca w nawiasy `(?!` *oraz* `)` *nie powoduje powstania odwołania wstecznego, a powoduje, że poszukiwanie następnego dopasowanie rozpocznie się przed niedopasowanym właśnie fragmentem*

Wyrażenia regularne

Przykład: ***wymaganie: szukamy słowa, które nie zaczyna się od „za”***

Wzorzec:

```
"\b(?!za)\w+\b"
```

Wyrażenia regularne

Zasada: nazwane odwołanie wsteczne

ujęcie fragmentu wzorca w nawiasy **(?<nazwa>** *oraz* **)** *powoduje powstania odwołania wstecznego, do którego można odwołać się poprzez nazwę podaną w nawiasach* **<** *i* **>**

Wyrażenia regularne

Przykład: *wyrażenie regularne, które sprawdza, czy pewien łańcuch jest poprawnym adresem e-mail oraz może posłużyć do wygodnego wyłuskania nazwanych podłańcuchów*

Wzorzec:

```
"(?<user>[^@]+)@(?<host>.+)"
```