Лабораторная работа №13 (3-й семестр)

> Измерять продуктивность программирования подсчетом строк кода — это так же, как оценивать постройку самолета по его весу.
>
> — *Bill Gates*

Теоретические сведения

Регуля́рные выраже́ния (англ. regular expressions) — формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов (символов-джокеров, англ. wildcard characters). По сути это строка-образец (англ. pattern, по-русски её часто называют «шаблоном», «маской»), состоящая из символов и метасимволов и задающая правило поиска.

Регулярные выражения используются некоторыми текстовыми редакторами и утилитами для поиска и подстановки текста. Например, при помощи регулярных выражений можно задать шаблоны, позволяющие:

- найти все последовательности символов «кот» в любом контексте, как то: «кот», «котлета», «терракотовый»;
- найти отдельно стоящее слово «кот» и заменить его на «кошка»;
  найти слово «кот», которому предшествует слово «персидский» или «чеширский»;
- убрать из текста все предложения, в которых упоминается слово кот или кошка.

Регулярные выражения позволяют задавать и гораздо более сложные шаблоны поиска или замены.

Обычные символы (литералы) и специальные символы (метасимволы)

Большинство символов в регулярном выражении представляют сами себя за исключением специальных символов [ ] \ / ^ $. | ? * + ( ) { }, которые могут быть предварены символом \ (обратная косая черта) («экранированы», «защищены») для представления их самих в качестве символов текста. Можно экранировать целую последовательность символов, заключив её между \Q и \E.

Метасимвол **.** (точка) означает один любой символ, но в некоторых реализациях исключая символ новой строки.

Символьные классы (наборы символов)

Набор символов в квадратных скобках [ ] именуется символьным классом и позволяет указать интерпретатору регулярных выражений, что на данном месте в строке может стоять один из перечисленных символов. В частности, [абв] задаёт возможность появления в тексте одного из трёх указанных символов, а [1234567890] задаёт соответствие одной из цифр. Возможно указание диапазонов символов: например, [А-Яа-я] соответствует всем буквам русского алфавита, за исключением букв «Ё» и «ё».

Если требуется указать символы, которые не входят в указанный набор, то используют символ ^ внутри квадратных скобок, например [^0-9] означает любой символ, кроме цифр.

Некоторые символьные классы можно заменить специальными метасимволами:

Табл. 1 — Некоторые символьные классы

Символ	Эквивалент	Соответствие
\d	[0-9]	Цифровой символ
\D	[^0-9]	Нецифровой символ
\s	[\f\n\r\t\v]	Пробельный символ
\S	[^ \f\n\r\t\v]	Непробельный символ
\w	[[:word:]]	Буквенный или цифровой символ или знак подчёркивания
\W	[^[:word:]]	Любой символ, кроме буквенного или цифрового символа или знака подчёркивания

Позиция внутри строки. Следующие символы позволяют позиционировать регулярное выражение относительно элементов текста: начала и конца строки, границ слова.

Табл. 2 — Метасимволы для позиционирования регулярного выражения в тексте

Представление	Позиция	Пример
^	Начало строки	^a
$	Конец строки	a$
\b	Граница слова	a\b
		\ba
\B	Не граница слова	\Ba\B
\G	Предыдущий успешный поиск	\Ga

Квантификация (поиск последовательностей). Квантификатор после символа, символьного класса или группы определяет, сколько раз предшествующее выражение может встречаться. Следует учитывать, что квантификатор может относиться более чем к одному символу в регулярном выражении, только если это символьный класс или группа.

Табл. 3 — Квантификаторы

Представление	Число повторений		Пример	Соответствие
{n}	Ровно n раз		colou{3}r	colouuur
{m,n}	От m до n включительно		colou{2,4}r	colouur, colouuur, colouuuur
{m,}	Не менее m		colou{2,}r	colouur, colouuur, colouuuur и т. д.
{,n}	Не более n		colou{,3}r	color, colour, colouur, colouuur
?	Ноль или одно	{0,1}	colou?r	color, colour
*	Ноль или более	{0,}	colou*r	color, colour, colouur и т. д.
+	Одно или более	{1,}	colou+r	colour, colouur и т. д. (но не color)

Часто используется последовательность .* для обозначения любого количества любых символов между двумя частями регулярного выражения.

Символьные классы в сочетании с квантификаторами позволяют устанавливать соответствия с реальными текстами. Например, столбцами цифр, телефонами, почтовыми адресами, элементами HTML-разметки и др.

Если символы { } не образуют квантификатор, их специальное значение игнорируется.

Жадная и ленивая квантификация

В некоторых реализациях квантификаторам в регулярных выражениях соответствует максимально длинная строка из возможных (квантификаторы являются жадными, англ. greedy). Это может оказаться значительной проблемой. Например, часто ожидают, что выражение (<.*>) найдёт в тексте теги HTML. Однако, если в тексте есть более одного HTML-тега, то этому выражению соответствует целиком строка, содержащая множество тегов.

Выражение (<.*>) соответствует строке, содержащей несколько тегов HTML-разметки, целиком.

Википедия — свободная энциклопедия, в которой каждый может изменить или дополнить любую статью.

Чтобы выделить отдельные теги, можно применить ленивую версию этого выражения: (<.*?>) Ей соответствует не вся показанная выше строка, а отдельные теги (выделены цветом):

Википедия — свободная энциклопедия, в которой каждый может изменить или дополнить любую статью.

Использование ленивых квантификаторов может повлечь за собой обратную проблему, когда выражению соответствует слишком короткая, в частности, пустая строка.

Класс wxRegEx. Класс предоставляет инструменты для работы с регулярными выражениями. Класс находится в библиотетке «wx/regex.h». Основные члены класса приведены в табл.

Табл. - некторые методы класса wxRegEx

Метод	**Описание**
wxRegEx(const wxString& expr, int flags = wxRE_DEFAULT)	Создает и компилирует регулярное выражение,
bool Compile(const wxString& pattern, int flags = wxRE_DEFAULT)	Компилирует регулярное выражение, возвращает истину в случае успеха
bool IsValid()	Возвращает истину, если регулярное выражение составлено без синтаксических ошибок
bool GetMatch(size_t* start, size_t* len, size_t index = 0) const	Возвращает индекс и длину вхождения выражения
size_t GetMatchCount() const	Возвращает длинну массива вхождений
bool Matches(const wxChar* text, int flags = 0) const	Проверяет на наличие соответсвий в введенном тексте
int ReplaceAll(wxString* text, const wxString& replacement) const	Заменяет в строке text все вхождения регулярного выражения на replacement

```
wxString text=_T("trololo ghjhf jhgfjhgf kghkhkgv a@a.com.ua");

    wxRegEx re(_(".+"));
    if (re.IsValid())
    {
        wxMessageBox(_T("Valid"));
        if (re.Matches(text))
        {
        wxString result =  re.GetMatch(text, 0);
        wxMessageBox(result);
        }
    }
```

Регулярные выражения

Якоря

^	Начало строки +
\A	Начало текста +
$	Конец строки +
\Z	Конец текста +
\b	Граница слова +
\B	Не граница слова +
\<	Начало слова
\>	Конец слова

Символьные классы

\c	Управляющий символ
\s	Пробел
\S	Не пробел
\d	Цифра
\D	Не цифра
\w	Слово
\W	Не слово
\xhh	Шестнадцатиричный символ hh
\Oxxx	Восьмиричный символ xxx

Символьные классы POSIX

[:upper:]	Буквы в верхнем регистре
[:lower:]	Буквы в нижнем регистре
[:alpha:]	Все буквы
[:alnum:]	Буквы и цифры
[:digit:]	Цифры
[:xdigit:]	Шестнадцатиричные цифры
[:punct:]	Пунктуация
[:blank:]	Пробел и табуляция
[:space:]	Пустые символы
[:cntrl:]	Управляющие символы
[:graph:]	Печатные символы
[:print:]	Печатные символы и пробелы
[:word:]	Буквы, цифры и подчеркивание

Утверждения

?=	Вперед смотрящее +
?!	Отрицательное вперед смотрящее +
?<=	Назад смотрящее +
?!= или ?	Отрицательное назад смотрящее +
?>	Однократное подвыражение
?()	Условие [если, то]
?()|	Условие [если, то, а иначе]
?#	Комментарий

Примечание *Отмеченное + работает в большинстве языков программирования.*

Образцы шаблонов

([A-Za-z0-9-]+)	Буквы, числа и знаки переноса
(\d{1,2}\/\d{1,2}\/\d{4})	Дата (напр., 21/3/2006)
([^\s]+(?=\.(jpg|gif|png))\.\2)	Имя файла jpg, gif или png
(^[1-9]{1}$\|^[1-4]{1}[0-9]{1}$|^50$)	Любое число от 1 до 50 включительно
(#?([A-Fa-f0-9]){3}(([A-Fa-f0-9]){3})?)	Шестнадцатиричный код цвета
((?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,15})	От 8 до 15 символов с минимум одной цифрой, одной заглавной и одной строчной буквой (полезно для паролей).
(\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,6})	Адрес email
(\<(/?[^\>]+)\>)	HTML теги

Примечание *Эти шаблоны предназначены для ознакомительных целей и основательно не проверялись. Используйте их с осторожностью и предварительно тестируйте.*

Кванторы

*	0 или больше +
*?	0 или больше, нежадный +
+	1 или больше +
+?	1 или больше, нежадный +
?	0 или 1 +
??	0 или 1, нежадный +
{3}	Ровно 3 +
{3,}	3 или больше +
{3,5}	3, 4 или 5 +
{3,5}?	3, 4 или 5, нежадный +

Специальные символы

\	Экранирующий символ +
\n	Новая строка +
\r	Возврат каретки +
\t	Табуляция +
\v	Вертикальная табуляция +
\f	Новая страница +
\a	Звуковой сигнал
[\b]	Возврат на один символ
\e	Escape-символ
\N{name}	Именованый символ

Подстановка строк

$n	n-ая непассивная группа
$2	«xyz» в /^(abc(xyz))$/
$1	«xyz» в /^(?:abc)(xyz)$/
$`	Перед найденной строкой
$'	После найденной строки
$+	Последняя найденная строка
$&	Найденная строка целиком
$_	Исходный текст целиком
$$	Символ «$»

Диапазоны

.	Любой символ, кроме переноса строки (\n) +
(a|b)	a или b +
(...)	Группа +
(?:...)	Пассивная группа +
[abc]	Диапазон (a или b или c) +
[^abc]	Не a, не b и не c +
[a-q]	Буква между a и q +
[A-Q]	Буква в верхнем регистре между A и Q +
[0-7]	Цифра между 0 и 7 +
\n	n-ая группа/подшаблон +

Примечание *Диапазоны включают граничные значения.*

Модификаторы шаблонов

g	Глобальный поиск
i	Регистронезависимый шаблон
m	Многострочный текст
s	Считать текст одной строкой
x	Разрешить комментарии и пробелы в шаблоне
e	Выполнение подстановки
U	Нежадный шаблон

Мета-символы (экранируются)

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

Эта таблица доступна на www.exlab.net
Англоязычный оригинал на AddedBytes.com