

Lab 1: lexical analysis (ilab)

Igor Gorban

3.10.18

mipt.igor.gorban@gmail.com

Решаемая задача

Написать лексический анализатор для языка COOL.

Для этого необходимо использовать подготовленную инфраструктуру сборки и тестирования.

Репозиторий:

https://gitlab.com/mipt.igor.gorban/pa1_lexer.git

Лексический анализатор находится в файле ***cool.flex***.

Для того чтобы понять как его писать необходимо заглянуть в файлы:

1. handouts/PA2.pdf (самостоятельно)
2. handouts/cool-manual.pdf (самостоятельно)
3. include/cool-parse.h + 116 (Tokens)

Так же придется разобраться как работает flex и научиться регулярным выражениям.
– Этим сегодня и будем заниматься.

В помощь – “cool-manual.pdf: figure 1. Cool syntax” и “cool-parse.h:line 116”(список токенов).

Регулярные выражения, краткая таблица.

.	Один любой символ (кроме \$)	[ab].	?	Проверка ноль или один раз (есть ли совпадение)?	(a)? [abc]?
^	Отрицание последовательности символов (если в скобках). Начало строки	[^abc] ^a	[]	Набор символов, с которыми ищется совпадение	[abc] [a-c]
\$	Символ окончания ввода	[abc]\$	\	Специальный символ	[\t\n\r\f\v]
*	Повторение последовательности ноль и более раз	[abc]*		«Или» для выбора группы символов	(a b c)
+	То же что и *, но один и более раз	[abc]+	()	Объединение группы символов	(abc)
ε	Эпсилон - пустой символ. (Обычно не пишется)	^\$	{m,n}	Не менее m и не более n повторений	{2,4}r

Где используются регулярные выражения?

Имплементированы в языках Perl, Java, PHP, JavaScript, Python, Ruby, Lua, C++, Delphi, D.

Регулярные выражения широко используются в UNIX и UNIX-подобных утилитах, например в `expr`, `awk`, `Emacs`, `vi(m)`, `(f)lex` и `grep`.

`Sed`: потоковый текстовый редактор. Повсеместно используется для поиска и замены регулярных выражений, однако является более мощным инструментом.

Регулярные выражения, ссылки.

Построители ДКА по регулярному выражению

<https://cyberzhg.github.io/toolbox/> ($-(0|1)^*\backslash(0|1)^+$)

<http://ivanzuzak.info/noam/webapps/fsm2regex/>

($(-+)(0+1+)^*\backslash(0+1+)^*$)

Regular expression online : <https://regex101.com/> ($-(0|1)^*\backslash(0|1)^+$)

-100010.001

-.001

Если заинтересовало – показать шахматы на sed:

<https://github.com/bolknote/SedChess>

Flex (Fast Lexical Analyzer) Определение.

%{

Код, копируемый *почти* в начало сгенерированной программы

%}

Блок определений (декларации и настройки параметров)

%%

Блок правил (шаблон и соответствующий ему код).

Шаблон задается регулярным выражением.

В начале строки - нет пробелов!

%%

Блок кода - основная программа, если вы не пишете компилятор COOL

Flex переменные.

- `ytext` – строковая переменная. Заполняется текстом токена, распознанного по регулярному выражению.
- `yylval` – переменная, определяемая парсером (bison). Может быть структурой (или объединением). Цель – передать в парсер значение токена.

Пример:

```
[\\] { cool_yylval.error_msg = ytext; return ERROR; }
```

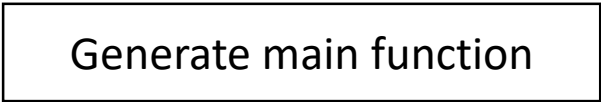
В **COOL** `yylval` определяется так (Смотри `cool-parse.h`):

```
#define yylval cool_yylval
```

В задании необходимо использовать поля `boolean`, `symbol`, `error_msg`.

- <http://dinosaur.compilertools.net/flex/manpage.html> - MULTIPLE INPUT BUFFERS

Flex примеры

```
%{  
#include <stdio.h>  
%}  
%option main  
%%  
[a-z]+ {printf("VARIABLE %s \n",yytext);}   
[0-9]+ {printf("NUMBER %s \n",yytext);}   
[+] {printf("TOKEN + \n");}  
[*] {printf("TOKEN * \n");}  
[=] {printf("TOKEN = \n");}[ \t\r\n]  
%%
```

```
%{  
Int num_lines = 0, num_chars = 0  
%}  
%option noyywrap  
%%  
\n      ++num_lines; ++num_chars;  
.      ++num_chars;  
%%  
Int main()  
{  
    yylex();  
    printf("lines = %d, chars = %d\n",  
          num_lines, num_chars);  
}
```


Flex %x

Во flex существует возможность определять «состояния». Состояния задаются в блоке определений. Для определения состояния необходимо написать «%x <ИМЯ_СОСТОЯНИЯ>». Из состояния в состояние можно переходить в коде. Однако воспринимают его шаблоны правил. По умолчанию задано одно начальное состояние - INITIAL. В нем по-умолчанию программа и находится. Если состояние не задано - считается, что это INITIAL.

Пример (рабочий код):

...

%x COMMENT

%%

...

```
<COMMENT>"\*\" {  
    in_comment--;  
    if (in_comment == 0) BEGIN(INITIAL);
```

Flex <<EOF>>

Всякий раз, когда сканер flex достигает конца входного файла, он готов проматчиться с паттерном << EOF >> .

Пример:

...

```
<STRING><<EOF>> {cool_yylval.error_msg = "EOF in string constant»;}  
<SMALL_COMMENT><<EOF>> BEGIN(INITIAL);
```

Резюме - <<EOF>> используется как «символ» конца файла.

Cool.flex

Относящиеся к строке переменные:

```
#define MAX_STR_CONST 1025  
char string_buf[MAX_STR_CONST];  
char *string_buf_ptr;
```

На основании константы MAX_STR_CONST построены тесты. То есть считается, что строка в кавычках не может превышать это количество!

См. cool-manual.pdf - 7.1. Constants

```
Пример:  [\"] { string_buf_ptr = string_buf; BEGIN(STRING);}  
extern int curr_lineno;
```

Конечно же в этой переменной должна храниться строка в файле.

```
Пример: <COMMENT>\n ++curr_lineno;
```

Лексическая структура языка Cool

vsplit examples/cool/examples/sort_list.cl

Лексические единицы языка Cool - это <целые числа>, <идентификаторы типов>, <идентификаторы объектов>, <специальные обозначения>, <строки>, <ключевые слова> и <пробельные символы>.

<Целые числа> - это непустые строки цифр от 0 до 9.

<Идентификаторы> - это строки (не совпадающие с ключевыми словами), состоящие из буквы, цифры и символ подчеркивания.

<Идентификаторы типов> начинаются с заглавной буквы.

<Идентификаторы объектов> начинаются с буквы нижнего регистра.

Есть два других идентификатора: `self` и `SELF_TYPE`, которые обрабатываются специально в Cool, но не рассматриваются как ключевые слова.

Лексическая структура языка Cool

Строки заключены в двойные кавычки «...».

Внутри строки последовательность '\с' обозначает символ 'с', за исключением следующего:

\b - возврат каретки \t – табуляция \n - перевод на новую строку \f - конец страницы

Для того, чтобы поместить в строку несколько строк - можно использовать экранирование.

Помещение в строку неэкранированного перевода строки - недопустимо:

```
"This \
```

```
is OK "
```

```
"This is not
```

```
OK "
```

<Строка> не может содержать EOF(конец файла).

Строка не может содержать ноль (символ \0).

Любой другой символ может содержаться в строке.

Файл не может закончиться, если строка не закрыта.

Лексическая структура языка Cool

<Комментарии> В Cool есть две формы комментариев.

Любые символы между двумя штрихами "--" и следующей новой строкой (или EOF) рассматриваются как комментарии.

Комментарии также могут быть написаны путем включения текст в (* ... *). Этот тип комментария может быть вложенным(!).

Файл не может закончиться, если строка не закрыта.

Ключевые слова cool: class, else, false, fi, if, in, inherits, isvoid, let, loop, pool, then, while, case, esac, new, of, not, true.

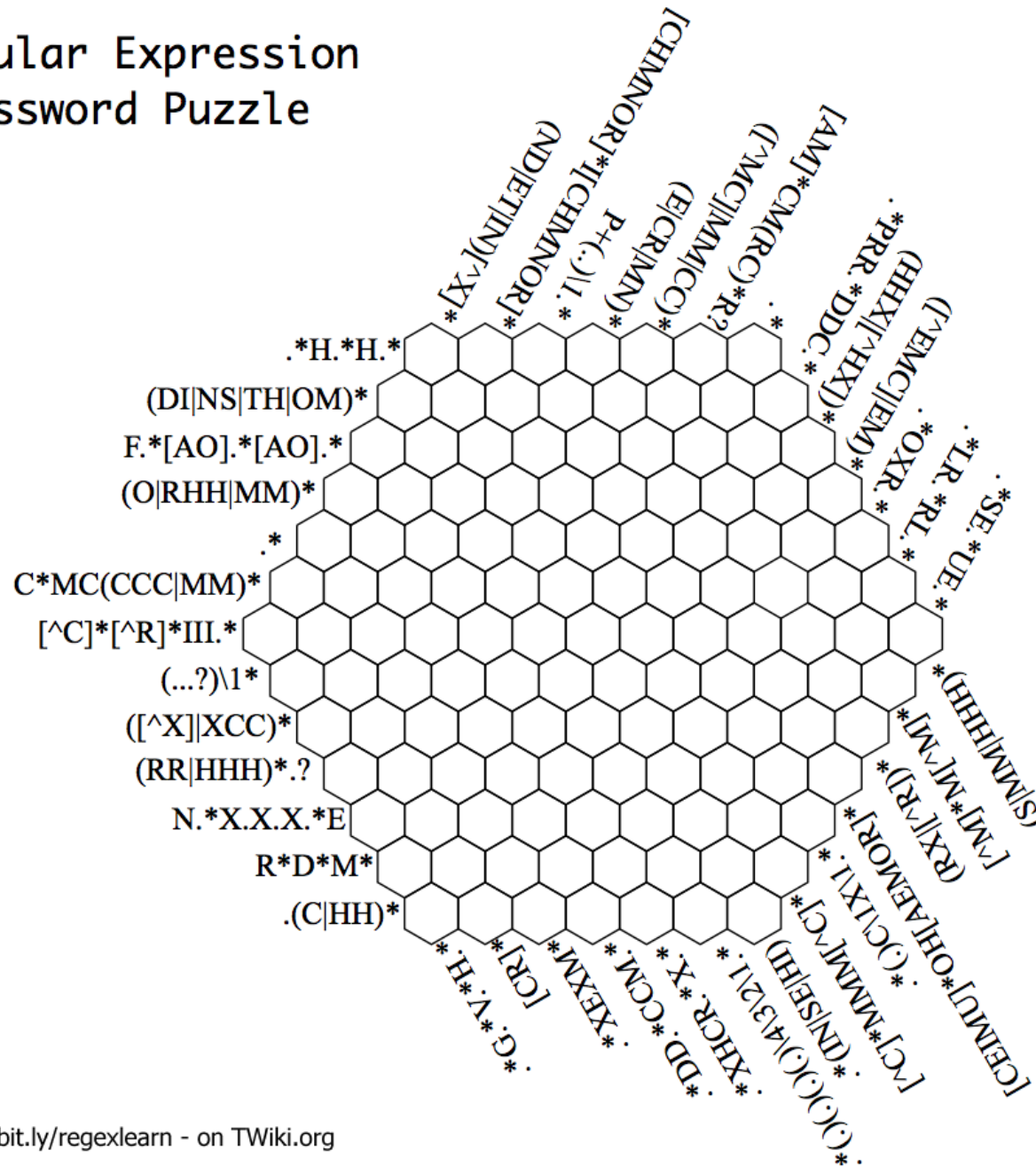
Все ключевые слова, кроме true и false, нечувствительны к регистру(!). Однако, чтобы поддерживать логику остальных ключевых слов - только первая буква <true> и <false> должна быть строчной, остальные буквы могут быть как в верхнем, так и в нижнем регистре (tRUe - ok, True - not ok).

Regular Expression Crossword Puzzle

ИСТОЧНИК:

<http://twiki.org/cgi-bin/view/Codev/TWikiPresentation2013x03x07>

Еще больше на <https://regexcrossword.com/>



Links

- Flex:
 - Статья на русском:
 - <http://rus-linux.net/lib.php?name=/MyLDP/algol/lex-yacc-howto.html>
 - <https://www.gnu.org/software/flex/>
 - <http://dinosaur.compilertools.net/flex/index.html>
 - ftp://ftp.gnu.org/old-gnu/Manuals/flex-2.5.4/html_mono/flex.html