Санкт-Петербургский государственный политехнический университет

Факультет технической кибернетики

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №1**

«Построение лексических анализаторов   
(утилита lex)»

Работу выполнил студент группы № 4081/12

Дорофеев Юрий Владимирович

Работу принял преподаватель \_\_\_\_\_\_\_\_\_\_\_\_

Цыган Владимир Николаевич

г. Санкт-Петербург

2012

1. **Цель работы**

Изучение и получение навыков применения утилиты lex для генерирования лексических анализаторов.

1. **Программа работы**

**2.1. Введение**

**Пример 1.** Копирование в выходной поток все кроме табуляций в начале строки

Код программы ex1.l:

%%   
^[ \t]+     ;   
%%   
#include "yy.c"

Код программы yy.c :

int yywrap() { return( 1 ); }

main()

{

while( yylex() );

}

На первом этапе применяется lex-транслятор. Запуск утилиты производится из командной строки в следующем виде:

dorofeev@dorofeev-VirtualBox:~/lablex$ /usr/local/flex/bin/flex -w ex1.l

На втором этапе осуществляется компиляция программы лексического анализатора:

dorofeev@dorofeev-VirtualBox:~/lablex$ gcc -o scaner.out lex.yy.c

Проверка работоспособности лексического анализатора осуществляется стандартным образом:

dorofeev@dorofeev-VirtualBox:~/lablex$ ./scaner.out <text.in> text.out

Если во входном файле text.in строка начинается с табуляции, то в выходной файл text.out поле трансляции, компиляции и запуска программы запишется эта же строка, но без табуляции.

Файл text.in :

dorofeev@dorofeev-VirtualBox:~/lablex$ cat text.in

lfld;fds

;sld;flsd

fdsdfs sdffds fsd

Файл text.out :

dorofeev@dorofeev-VirtualBox:~/lablex$ cat text.out

lfld;fds

;sld;flsd

fdsdfs sdffds fsd

**2.2. Действия**

Действие - это заданный на языке C фрагмент кода, который выполняется при выявлении соответствующего шаблона. Обычно действия содержат такие операции, как преобразование выявленной строки - лексемы, возвращение типа лексемы процедуре грамматического разбора, или сбор статистики о входном потоке.

В следующем примере распознаются все возможные литеры, за счет чего правило по умолчанию не используется. Полезная работа этой программы сводится к подсчету количества строк.

**Пример 2.** Распознание всех возможных литеры и подсчет количества строк

Код программы ex2.l:

int lineno = 0;   
%%   
\n     lineno++;   
.       ;   
%%   
yywrap()                 
{   
        printf( "%d lines\n", lineno );   
        return( 1 );   
}   
main()   
{   
    while( yylex() );   
}

Функция yywrap(), определяемая пользователем, вызывается при завершении входного потока. Чтобы лексический анализатор закончил работу, функция yywrap() должна вернуть ненулевой результат.

Содержимое text.in:

dorofeev@dorofeev-VirtualBox:~/lablex/2$ cat text.in

first

second

third

fifth

Содержимое text.out:

dorofeev@dorofeev-VirtualBox:~/lablex/2$ cat text.out

5 lines

Входная последовательность, будучи распознана некоторым правилом, сохраняется в массиве yytext, а ее длина записывается в переменную yyleng.

**Пример 3.** Подсчет последовательностей, которые обозначают знаковые целые числа

Каждый раз при обнаружении такой последовательности выводится текущее значение счетчика и текст лексемы:

Код программы ex3.l :

%{

int count = 0;

%}

%%

[-+]?[0-9]+ {

count++;

printf( "%d %s\n", count, yytext );

}

%%

int yywrap() { return( 1 ); }

main()

{

while( yylex() );

}

Содержимое text.in:

dorofeev@dorofeev-VirtualBox:~/lablex/3$ cat text.in

+5-554-5445

-546546554

Содержимое text.out:

dorofeev@dorofeev-VirtualBox:~/lablex/3$ cat text.out

1 +5

2 -554

3 -5445

4 -546546554

**Пример 4.** Перепись идентификаторов и беззнаковых чисел в выходной поток по одному на строке

Макрокоманда ECHO производит вывод содержимого yytext.

Код программы ex4.l:

%%   
[0-9]+         |   
[a-zA-Z]+     { ECHO; printf( "\n" ); }   
.    |    \n     ;   
%%   
int yywrap() { return( 1 ); }

main()

{

while( yylex() ); }

Литера «|»' вместо действия означает то же действие, что и для следующего правила''.

Содержимое text.in:

dorofeev@dorofeev-VirtualBox:~/lablex/4$ cat text.in

Hdfhfgh56fsdfgt87564ffff

Содержимое text.out:

dorofeev@dorofeev-VirtualBox:~/lablex/4$ cat text.out

Hdfhfgh

56

fsdfgt

87564

ffff

Как видно из результатов, программа работает верно: во входящем потоке выделяются буквенные идентификаторы и беззнаковые числа.

**Пример 5.** Использование yyleng (программа подсчета идентификаторов по длине, в диапазоне длины от 1 до 40)

Код программы ex5.l:

%{

int len[40];   
    int i;

%}

%%   
        {   
            for( i = 0; i < 40; i++ )   
                    len[i] = 0;   
        }   
[a-z]+         len[yyleng]++;   
.    |    \n     ;   
%%   
yywrap()   
{   
    for( i = 0; i < 40; i++ )   
            if( len[i] > 0 ) printf( "%5d%10d\n", i, len[i] );   
    return( 1 );   
}   
main()   
{   
    while( yylex() );   
}

Содержимое text.in:

dorofeev@dorofeev-VirtualBox:~/lablex/5$ cat text.in

fdsg

a

fg

fdfff

dafgr

Содержимое text.out:

dorofeev@dorofeev-VirtualBox:~/lablex/5$ cat text.out

1 1

2 1

4 1

5 2

Вывод программы осуществлен в виде гистограммы.

Когда очередная лексема записывается в массив yytext, предыдущее содержимое yytext теряется. Функция yymore() временно отключает режим "перезаписи'' для следующего (одного) сопоставления, т.е. литеры следующей лексемы будут добавлены к текущему содержимому yytext. Функция yyless(n) сокращает содержимое yytext до n первых литер, возвращая остальные во входной поток.

**Пример 6.** Удаление первых пяти букв из любого слова, которое начинается с большой буквы и содержит более пяти литер:

 Код программы ex6.l:

%{   
#define limit 5   
%}   
%%   
[A-Z][a-z]\*         if( yyleng > limit ) yyless( limit );   
[a-z]\*                 ECHO;   
%%   
#include "yy.c"

Содержимое text.in:

dorofeev@dorofeev-VirtualBox:~/lablex/6$ cat text.in

Fgktly

GFGH

gfgf

abcdefg

tABCdefgr

Содержимое text.out:

dorofeev@dorofeev-VirtualBox:~/lablex/6$ cat text.out

y

gfgf

abcdefg

tr

Как видно из результатов программы, при задании слова, написанного прописными буквами (GFGH), оно полностью удаляется, также удаляются прописные буквы, которые расположены в середине слова и если после последней прописной буквы идут строчные, то произойдет удаление 4-х из них (tABCdefgr -> tr).

Пользователь также может обращаться к функциям низкоуровневого ввода-вывода, которые используются лексическим анализатором:

* input() - чтение следующей литеры входного потока (в конце потока считывается null-литера);
* output(c) - запись литеры "c'' в выходной поток;
* unput(c) - "запись'' литеры во входной поток.

**Пример 7.** Поиск конца комментария, заданного в стиле языка C

Код программы ex7.l:

%{   
void skip\_comments();   
%}   
D         [0-9]   
H         [0-9A-Fa-f]   
L         [A-Za-z]   
%%   
{L}({L}|{D})\*         printf( "ident: %s\n", yytext );   
0{H}+(H|h)?     |   
{D}{H}\*(H|h)         printf( "hex: %s\n", yytext );   
{D}+                       printf( "decimal: %s\n", yytext );   
"/\*"                         skip\_comments();   
.                             ;   
%%   
void skip\_comments()   
{   
    char c = '\*';   
    while( c != '/' ) {   
        while( input() != '\*' );   
            c = input();   
            if( c != '/' ) unput (c);   
    }   
}   
#include "yy.c"

В примере определены правила для распознавания десятичных и 16-ричных констант (в стиле A86) и идентификаторов. Для сокращения записи этих правил в разделе определений заданы макроопределения шаблонов, обозначающих буквы, десятичные и 16-ричные цифры.

Если 16-ричное число начинается с буквы, то для того, чтобы оно не воспринималось, как идентификатор необходимо перед ним поставить "0", а в конце "h".

Если 16-ричное число начинается с цифры [0-9], то 0 ставить необязательно

Содержимое text.in:

dorofeev@dorofeev-VirtualBox:~/lablex/7$ cat text.in

/\*True\*/

id1 7

id2 3Fh

8

3Fh

0F5h

/\*False\*/

id4Bh

A4h

id4 A4h

Содержимое text.out:

dorofeev@dorofeev-VirtualBox:~/lablex/7$ cat text.out

ident: id1

decimal: 7

ident: id2

hex: 3Fh

decimal: 8

hex: 03Fh

hex: 0F5h

ident: id4Bh

ident: A4h

ident: id4

ident: A4h

Из результатов видно, что программа работает правильно: удаляет комментарии, определяет тип числа – 16-ричное или 10-ричное, определяет идентификаторы.

**Пример 8.** Реверсирование идентификаторов с префиксом '@'

Код программы ex8.l:

%{

    int i;

%}  
%%   
\@[A-Za-z]+  {   
                 for( i = 1; i < yyleng; i++ )   
                      unput( yytext[i] );   
              }   
%%   
#include "yy.c"

Для правильной работы программы, необходимо использовать ключ -l при ее трансляции:

dorofeev@dorofeev-VirtualBox:~/lablex$ /usr/local/flex/bin/flex –wl ex8.l

Данный ключ служит для максимальной совместимости с оригинальным LEX. Без него происходит реверсирование не полного слова, т.к. вероятнее всего для хранения считываемых символов используются одни и те же ячейки памяти.

Содержимое text.in:

dorofeev@dorofeev-VirtualBox:~/lablex/8$ cat text.in

@Gfdhrht

@hgfhfg

@21hht

abcd

Содержимое text.out:

dorofeev@dorofeev-VirtualBox:~/lablex/8$ cat text.out

thrhdfG

gfhfhg

@21hht

abcd

**2.3. Управление правилами**

Возможности управления набором правил:

* Разрешение двусмысленностей;
* Стартовые условия;
* Действие REJECT.

**2.3.1. Разрешение двусмысленностей**

Если при поиске лексемы входная последовательность может быть распознана несколькими шаблонами, то набор правил двусмысленный. В этой ситуации правило выбирается по следующей схеме:

1. Предпочтение отдается соответствию большей длины;
2. Если одна и та же последовательность соответствует нескольким правилам, предпочтение отдается тому правилу, которое задано раньше других.

**Пример 9.**

Код программы ex9.l:

%%   
read         { printf( "operation: " ); ECHO; }   
[a-z]+       { printf( "identifier: " ); ECHO; }   
%%   
#include "yy.c"

Содержимое text.in:

dorofeev@dorofeev-VirtualBox:~/lablex/9$ cat text.in

ready

read

readread

Содержимое text.out:

dorofeev@dorofeev-VirtualBox:~/lablex/9$ cat text.out

identifier: ready - Предпочтение отдается соответствию большей длины

operation: read - Оба правила распознают одинаковое число литер - 4, но будет выбрано первое правило, т.к. оно задано раньше

identifier: readread

Для правил с выражениями типа ".\*'' принцип предпочтения наиболее длинного соответствия опасен. Например, для выявления строк в одиночных кавычках может показаться подходящим следующее решение:

**Пример 10.**

Код программы ex10.l:

**%%**   
**'.\*'             ;**   
**%%**   
**#include "yy.c"**

Содержимое text.in:

dorofeev@dorofeev-VirtualBox:~/lablex/10$ cat text.in

'first' here, 'second' there

Содержимое text.out:

dorofeev@dorofeev-VirtualBox:~/lablex/10$ cat text.out

there

Фактически это приглашение к поиску самой дальней закрывающей кавычки. Решение задачи предыдущего примера следующее:

**Пример 11.**

Код программы ex11.l:

%%   
'[^'\n]\*'             ;   
%%   
#include "yy.c"

Содержимое text.in:

dorofeev@dorofeev-VirtualBox:~/lablex/11$ cat text.in

'first' here, 'second' there

Содержимое text.out:

dorofeev@dorofeev-VirtualBox:~/lablex/11$ cat text.out

here, there

**2.3.2. Стартовые условия**

Стартовые условия позволяют изменить набор действующих правил в процессе работы лексического анализатора. Посредством стартовых условий лексический анализатор адаптируется к изменению контекста.

Изменение окружающей обстановки, в принципе, может быть учтено тремя способами:

* Применение флагов, когда меняются лишь некоторые правила;
* Использование стартовых условий, наложенных на правила;
* Переключение между несколькими программами лексических анализаторов.

Рассмотрим следующую задачу: копировать входной поток, заменяя "magic'' на "first'' в строках, начинающихся с "a'', или на "second'' в строках, начинающихся с "b'', или на "third'', если строка начинается с "c''.

**Пример 12-1.** Использование флагов

Код программы ex12\_1.l:

int flag;   
%%   
^a                 { flag = 'a'; ECHO; }   
^b                 { flag = 'b'; ECHO; }   
^c                 { flag = 'c'; ECHO; }   
\n                 { flag = 0 ; ECHO; }   
magic           { switch (flag)   
                            {   
                                case 'a': printf("<first>"); break;   
                                case 'b': printf("<second>"); break;   
                                case 'c': printf("<third>"); break;   
                                default : ECHO;   
                            }   
                    }   
%%   
#include "yy.c"

Содержимое text.in:

dorofeev@dorofeev-VirtualBox:~/lablex/12$ cat text.in

autor magic

add maggd

centr magic

b is magic

B magic

Содержимое text.out:

dorofeev@dorofeev-VirtualBox:~/lablex/12$ cat text.out

autor <first>

add maggd

centr <third>

b is <second>

B magic

Замена «magic» на другое слово, в зависимости от первого символа строки, происходит в структуре switch-case.

Первые четыре правила можно сократить. За счет этого код программы уменьшится:

**Пример 12-2.** Использование флагов

Код программы ex12\_2.l:

int flag;   
%%   
^[abc]     |   
\n            { flag = yytext[0] ; ECHO; }   
magic         { switch (flag)   
                            {   
                             case 'a': printf("<first>"); break;   
                             case 'b': printf("<second>"); break;   
                             case 'c': printf("<third>"); break;   
                             default : ECHO;   
                            }   
             }   
%%   
#include "yy.c"

При выполнении программы в выходном файле будет то же что и в примере 12\_1.

**Пример 13-1.** Использование стартовых условий

Код программы ex13\_1.l:

%START AA BB CC   
%%   
^a                         { ECHO; BEGIN AA; }   
^b                         { ECHO; BEGIN BB; }   
^c                         { ECHO; BEGIN CC; }   
\n                          { ECHO; BEGIN 0; }   
<AA>magic          printf( "<first>" );   
<BB>magic          printf( "<second>" );   
<CC>magic          printf( "<third>" );   
%%   
#include "yy.c"

Стартовые условия перечислены в секции определений. Если в начале строки считана буква а, b или с, то происходит установка требуемого стартового условия (АА, ВВ или СС соответственно) с помощью макрокоманды BEGIN.

В итоге при выполнении программы в выходном файле будет такое же содержимое, что и в примере 12-1.

Содержимое text.in:

dorofeev@dorofeev-VirtualBox:~/lablex/14$ cat text.in

autor magic

add maggd

centr magic

b is magic

B magic

Содержимое text.out:

dorofeev@dorofeev-VirtualBox:~/lablex/14$ cat text.out

autor <first>

add maggd

centr <third>

b is <second>

B magic

На уровне реализации стартовые условия представлены целыми значениями (в частности, исходное условие INITIAL обозначается также как 0). Это обстоятельство дает возможность производить вычисления над стартовыми условиями, а также проводить трассировку стартовых условий:

**Пример 13-2.** Использование стартовых условий (трассировка)

Код программы ex13\_2.l:

%S AA BB CC   
%{   
#define YY\_USER\_ACTION { printf("(%d)", YYSTATE); }   
%}   
%%   
^[abc]                        { ECHO; BEGIN( yytext[0]-'a'+1 ); }   
\n                               { ECHO; BEGIN( 0 ); }   
<AA>magic               printf( "<first>" );   
<BB>magic               printf( "<second>" );   
<CC>magic               printf( "<third>" );   
%%   
#include "yy.c"

Содержимое text.in:

dorofeev@dorofeev-VirtualBox:~/lablex/15$ cat text.in

autor magic

add maggd

centr magic

b is magic

B magic

Содержимое text.out:

dorofeev@dorofeev-VirtualBox:~/lablex/15$ cat text.out

(0)a(1)u(1)t(1)o(1)r(1) (1)<first>(1)

(0)a(1)d(1)d(1) (1)m(1)a(1)g(1)g(1)d(1)

(0)c(3)e(3)n(3)t(3)r(3) (3)<third>(3)

(0)b(2) (2)i(2)s(2) (2)<second>(2)

(0)B(0) (0)m(0)a(0)g(0)i(0)c(0)

Макрокоманда YYSTATE возвращает целочисленное значение текущего стартового условия. Исходное условие = 0 (оно устанавливается, если был перевод строки "\n"). Если в начале строки определена буква a, b или с, то происходит высчитывание значения стартового условия (если а, то 1, если b, то 2, если с, то 3). Из полученных результатов видно, что значения стартовых условий АА, ВВ и СС следующие: 1, 2 и 3. Таким образом, стартовым условиям присваивается значение, начиная с 1 в порядке их следования.

**2.3.3. Действие REJECT**

Логика лексического анализатора, сгенерированного Lex, направлена на выявление смежных последовательностей во входном потоке. Выявление перекрывающихся последовательностей требует программирования дополнительной логики.

**Пример 14-1.**

Код программы ex14\_1.l:

int s = 0, h = 0;   
%%   
she                  s++;   
he                    h++;   
(.\n)                 ;   
%%   
yywrap()   
{   
    printf( "she: %d times, he: %d times\n", s, h );   
    return( 1 );   
}   
main()   
{   
    while( yylex() );   
}

Содержимое text.in:

dorofeev@dorofeev-VirtualBox:~/lablex/16$ cat text.in

sheehe

she

heshe

Содержимое text.out:

dorofeev@dorofeev-VirtualBox:~/lablex/16$ cat text.out

e

she: 3 times, he: 2 times

Здесь запрограммирован подсчет экземпляров последовательностей "she'' и "he''. Анализатор не будет распознавать те экземпляры "he'', которые находятся внутри "she'', т.к. после распознавания "she'' все эти литеры уходят из входной последовательности.

Для выявления некоторых последовательностей внутри других последовательностей (также представляющих интерес) нужно:

* Вернуть принятую последовательность во входной поток;
* Исключить правило, которым была распознана эта последовательность;
* Возобновить сопоставление.

Первая фаза этого действия может быть реализована вызовом yyless(0), вторая - при помощи стартовых условий. Вместе с тем, имеется возможность задать это действие одной макрокомандой REJECT.

**Пример 14-2.** Применение REJECT

Код программы ex14\_2.l:

int s = 0, h = 0;   
%%   
she                      { s++; REJECT; }   
he                        { h++; REJECT; }   
(.\n)                     ;   
%%   
yywrap()   
{   
    printf( "she: %d times, he: %d times\n", s, h );   
    return( 1 );   
}   
main()   
{   
    while( yylex() );   
}

Содержимое in.txt:

dorofeev@dorofeev-VirtualBox:~/lablex/17$ cat text.in

sheehe

she

heshe

Содержимое out.txt:

dorofeev@dorofeev-VirtualBox:~/lablex/17$ cat text.out

sheehshhesh

she: 3 times, he: 5 times

После обнаружения "she'' увеличивается счетчик "s'' и правило отвергается, "she'' возвращается на вход. Предпринимается попытка заново сопоставить тот же ввод с оставшимися шаблонами. В этом примере можно учесть то, что "she'' включает в себя "he'', но не наоборот, и убрать REJECT из второго действия.

1. **Выводы**

В ходе работы были получены навыки работы утилитой для построения лексических анализаторов. Возможность написания правил и действий для анализатора в стиле языка **С** — очень удобно, в связи с популярностью этого языка. Также достаточно удобно, наличие в утилите **LEX** макрокоманд **ECHO**, **REJECT** и т. д. избавляющих программиста от составления дополнительных алгоритмов и уменьшающих размер исходного кода.