

Санкт-Петербургский политехнический университет Петра Великого
Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе

Дисциплина: Транслирующие системы

Тема: Программирование лексического разбора на языке *lex*

Выполнил студент гр. 43501/3
Преподаватель

Мальцев М.С.
Цыган В.Н.

Санкт-Петербург
19 марта 2019 г.

1 Цель работы

- Познакомиться с генератором программ лексической обработки текстов Lex.
- Выполнить трансляцию предложенных программ на языке Lex. Протестировать их работу.

2 Ход работы

2.1 Удаление пробелов и табуляций в начале строк

Рассмотрим программу, которая передает в выходной поток все литеры входного потока кроме пробелов и табуляций в начале строки.

```
1 %%  
2 ^[\t]+ ;  
3 %%  
4  
5 #ifndef yywrap  
6 int yywrap() { return 1; }  
7 #endif  
8  
9 main () { while (yylex()); }
```

Тестовые данные, подаваемые на вход, приведены ниже:

```
1 3 spaces  
2 2 tabs  
3 2 space + tab ;  
4 the end
```

В результате получили:

```
1 3 spaces  
2 2 tabs  
3 2 space + tab ;  
4 the end
```

Программа отработала корректно. Пробелы и символы табуляции были заменены.

2.2 Подсчет числа строк

Программа подсчитывающая количество строк:

```
1 int lineno = 0;  
2 %%  
3 \n lineno++;  
4 . ;  
5 %%  
6  
7 #ifndef yywrap  
8 int yywrap() { return( 1 ); }
```

```

9  #endif
10
11  main()
12  {
13      while( yylex() );
14      printf( "%d lines\n", lineno );
15  }

```

Тестовые данные, подаваемые на вход, приведены ниже:

```

1  1 string
2  string 2
3  3 string
4  string 4

```

В результате получили:

```

1  4 lines

```

Программа отработала корректно. Количество строк было подсчитано верно.

2.3 Подсчет и вывод знаковых целых чисел

Программа подсчитывающая и выводящая количество знаковых чисел:

```

1  %{
2  int count = 0;
3  %}
4
5  %%
6  [-+]?[0-9]+ {
7              count++;
8              printf( "%d %s\n", count, yytext );
9              }
10 %%
11
12 #include "yy.c"

```

Тестовые данные, подаваемые на вход, приведены ниже:

```

1  fkgnsdfngsd ladsmfka s d sdf kas      efgdf123
2  +123 dfg -987 sdfg sdf --456
3  456
4  123
5  sdfsd

```

В результате получили:

```

1  fkgnsdfngsd ladsmfka s d sdf kas      efgdf1 123
2
3  2 +123
4    dfg 3 -987
5    sdfg sdf -4 -456
6
7  5 456
8
9  6 123
10
11 sdfsd

```

Программа отработала корректно. Видно, что при обнаружении целого числа программа добавляет к нему индекс.

2.4 Вывод идентификаторов и беззнаковых целых чисел

Используемая программа:

```
1 %%  
2 [0-9]+ |  
3 [a-zA-Z]+ { ECHO; printf( "\n" ); }  
4 .|\n ;  
5 %%  
6  
7 #include "yy.c"
```

Тестовые данные, подаваемые на вход, приведены ниже:

```
1 jfdshf asdjewh23 123+ + 45324 +3425 -rg455 -x3 +4  
2 231 -3 --3 x2 x mk3 20 4
```

В результате получили:

```
1 jfdshf  
2 asdjewh  
3 23  
4 123  
5 45324  
6 3425  
7 rg  
8 455  
9 x  
10 3  
11 4  
12 231  
13 3  
14 3  
15 x  
16 2  
17 x  
18 mk  
19 3  
20 20  
21 4
```

Программа отработала корректно. Все идентификаторы и беззнаковые целые числа были отдельно выведены.

2.5 Подсчет и вывод гистограммы длин слов

Текст используемой программы:

```
1 int len[40], i;  
2  
3 %%  
4 {  
5     for( i = 0; i < 40; i++ )  
6         len[i] = 0;
```

```

7      }
8      [a-z]+ len[yyleng]++;
9      .|\n    ;
10    %%
11
12    #ifndef yywrap
13    int yywrap() { return 1; }
14    #endif
15
16    main()
17    {
18        while( yylex() );
19        for( i = 0; i < 40; i++ )
20            if( len[i] > 0 )
21                printf( "%5d%10d\n", i, len[i] );
22    }

```

Тестовые данные, подаваемые на вход, приведены ниже:

```

1 df g qg gtr gg allin yoloyolo
2 metro Exodus RTX

```

В результате получили:

1	1	1
2	2	3
3	3	1
4	5	3
5	8	1

Программа отработала корректно. Была выведена длина слов и их количество.

2.6 Дополнительное задание

Цепочки из 0 и 1 не больше, чем 2 единицы подряд.

Разработанная программа:

```

1 int is_true = 1;
2
3 %%
4 (1){3} { printf("11\n1");}
5 \n ;
6 %%
7
8 #ifndef yywrap
9 int yywrap() { return( 1 ); }
10 #endif
11
12 main() { while( yylex() ); }

```

Тестовые данные, подаваемые на вход, приведены ниже:

```

1 000010110111001100010
2 111000111000111100110

```

В результате получили:

```
1 00001011011
2 100110001011
3 100011
4 100011
5 1100110
```

Программа отработала корректно.

2.7 Вывод строки наискосок при помощи yyless

Текст используемой программы:

```
1 %%
2 (.)+ {
3     printf(">%s\n", yytext);
4     if (yyleng > 1) yyless(yyleng/2);
5 }
6 %%
7
8 /*
9 %%
10 (.)+ {
11     printf(">%s\n", yytext);
12     if (yyleng > 1) yyless(yyleng/2);
13 }
14 (.|\\n) ;
15 %%
16 */
17
18 #include "yy.c"
```

• Тест 1

Тестовые данные, подаваемые на вход, приведены ниже:

```
1 Lorem ipsum dolor sit amet, consectetur adipiscing elit.
```

В результате получили:

```
1 >Lorem ipsum dolor sit amet, consectetur adipiscing elit.
2 >consectetur adipiscing elit.
3 >ipiscing elit.
4 >g elit.
5 >lit.
6 >t.
7 >.
```

• Тест 2

Тестовые данные, подаваемые на вход, приведены ниже:

```
1 Nunc consectetur diam neque, at congue lectus congue euismod. Quisque
   faucibus non arcu eu lobortis. Aliquam mollis nisl nisl, ac pretium
   velit commodo in.
```

В результате получили:

```

1 >Nunc consectetur diam neque, at congue lectus congue euismod. Quisque
   faucibus non arcu eu lobortis. Aliquam mollis nisl nisl, ac pretium
   velit commodo in.
2 > non arcu eu lobortis. Aliquam mollis nisl nisl, ac pretium velit commodo
   in.
3 >isl nisl, ac pretium velit commodo in.
4 >m velit commodo in.
5 >mmodo in.
6 > in.
7 >n.
8 >.
9 >

```

Программа отработала корректно.

2.8 Макросы и ввод-вывод низкого уровня

Программа, приведенная ниже, идентифицирует буквы и числа в 10-м и 16-м фарматах. Программа игнорирует комментарии.

```

1 %{
2 void skip_comments();
3 %{
4
5 D [0-9]
6 H [0-9A-Fa-f]
7 L [_A-Za-z]
8
9 %%
10 {L}({L}|{D})* printf( "ident: %s\n", yytext );
11 0{H}+(H|h)? |
12 {D}{H}*(H|h) printf( "hex: %s\n", yytext );
13 {D}+ printf( "decimal: %s\n", yytext );
14 "/*" skip_comments();
15 . ;
16 %%
17
18 void skip_comments()
19 {
20     int c = '*'; /* not char! */
21
22     while( c != '/' ) {
23         while( input() != '*' );
24         c = input();
25         if( c != '/' )
26             unput (c);
27     }
28 }
29
30 #include "yy.c"

```

Тестовые данные, подаваемые на вход, приведены ниже:

```

1 int main()
2 {
3     unsigned int n;
4     unsigned long long factorial = 1;

```

```

5
6    /*
7    unsigned int x;
8    factorial
9    **/
10
11    int hexVar = 7Ah;
12
13    return 0;
14 }

```

В результате получили:

```

1 ident: int
2 ident: main
3
4
5 ident: unsigned
6 ident: int
7 ident: n
8
9 ident: unsigned
10 ident: long
11 ident: long
12 ident: factorial
13 decimal: 1
14
15
16
17
18 ident: int
19 ident: hexVar
20 hex: 7Ah
21
22
23 ident: return
24 decimal: 0

```

Программа отработала корректно.

2.9 Проверка конца входного потока при использовании input

Настоящая программа, является дополнением предыдущей. В качестве нововведения добавлена проверка конца входного потока при использовании input. Проблема предшествующей программа заключалась в том, что происходило заикливание при незакрытом комментарии. Код модифицированной программы представлен ниже:

```

1 %{
2 void skip_comments();
3 %{
4
5 D    [0-9]
6 H    [0-9A-Fa-f]
7 L    [_A-Za-z]
8

```



```

9  %%
10 {L}({L}|{D})*    printf( "ident: %s\n", yytext );
11 0{H}+(H|h)?      |
12 {D}{H}*(H|h)     printf( "hex: %s\n", yytext );
13 {D}+             printf( "decimal: %s\n", yytext );
14 "/"*            skip_comments();
15 .                ;
16 %%
17
18 void skip_comments()
19 {
20     int c = '*';    /* not char! */
21
22     do {
23         while ((c = input()) != '*' && c != EOF) ;
24         while ((c = input()) == '*') ;
25     } while (c != '/' && c != EOF);
26     if (c == EOF) {
27         fprintf(stderr, "?-EOF in comment\n");
28         exit(1);
29     }
30 }
31
32 #include "yy.c"

```

Тестовые данные, подаваемые на вход, приведены ниже:

```

1  int main()
2  {
3      unsigned int n;
4      unsigned long long factorial = 1;
5
6      /*
7      unsigned int x;
8      factorial
9      **/
10
11     int hexVar = 7Ah;
12     return 0;
13
14     /*
15         some text
16 }

```

В результате получили:

```

1  ident: int
2  ident: main
3
4
5  ident: unsigned
6  ident: int
7  ident: n
8
9  ident: unsigned
10 ident: long
11 ident: long
12 ident: factorial
13 decimal: 1
14
15

```

```

16
17
18 ident: int
19 ident: hexVar
20 hex: 7Ah
21
22 ident: return
23 decimal: 0

```

Программа отработала корректно.

2.10 Функция unput

Приведенная ниже программа производит реверсирование идентификаторов, начинающихся с символа @:

```

1      int    i, len;
2      char   *p;
3
4  %%
5  \@[A-Za-z]+ {
6              len = yyleng;
7              p = (char *)strdup(ytext);
8
9              for( i = 1; i < len; i++ )
10                 unput( p[i] );
11          }
12  %%
13
14 #include "yy.c"

```

Тестовые данные, подаваемые на вход, приведены ниже:

```

1 But I @must
2 @
3 explain @to you how
4 all this @mistaken idea
5 @of denouncing pleasure

```

В результате получили:

```

1 But I tsum
2 @
3 explain ot you how
4 all this nekatsim idea
5 fo denouncing pleasure

```

Программа отработала корректно.

2.11 Двусмысленный набор правил

Приведенная ниже программа демонстрирует выбор правила при двусмысленном наборе.

```

1  %%
2  read    { printf( "operation: " ); ECHO; }
3  [a-z]+  { printf( "identifier: " ); ECHO; }

```

```
4 %%  
5  
6 #include "yy.c"
```

Тестовые данные, подаваемые на вход, приведены ниже:

```
1 reader  
2 ready  
3 read  
4 forread  
5 red
```

В результате получили:

```
1 identifier: reader  
2 identifier: ready  
3 operation: read  
4 identifier: forread  
5 identifier: red
```

Программа отработала корректно.

2.12 Неправильный шаблон для распознавания строки в кавычках

Ниже приведена программа, которая демонстрирует неверное определение шаблона, в задаче выявления строк, не заключенных в кавычки.

```
1 %%  
2 '.*' ;  
3 %%  
4  
5 #include "yy.c"
```

Тестовые данные, подаваемые на вход, приведены ниже:

```
1 At 'vero' eos et 'accusamus' et  
2 iusto 'odio' dignissimos  
3 earum 'rerum hic tenetur  
4 maxime placeat facere' possimus
```

В результате получили:

```
1 At et  
2 iusto dignissimos  
3 earum 'rerum hic tenetur  
4 maxime placeat facere' possimus
```

Программа отработала некорректно, так как вместо того, чтобы исключить слова выделенные в кавычках она исключила и всё, что в пределах строки между выделенными словами находилось.

2.13 Правильный шаблон для распознавания строки в кавычках

Приведенная ниже программа демонстрирует работу правильного шаблона для распознавания строки в кавычках.

```
1 %%  
2 '[^'\n]*' ;  
3 %%  
4  
5 #include "yy.c"
```

Тестовые данные, подаваемые на вход, приведены ниже:

```
1 At 'vero' eos et 'accusamus' et  
2 iusto 'odio' dignissimos  
3 earum 'rerum hic tenetur  
4 maxime placeat facere ' possimus
```

В результате получили:

```
1 At eos et et  
2 iusto dignissimos  
3 earum 'rerum hic tenetur  
4 maxime placeat facere ' possimus
```

Программа отработала корректно.

2.14 Использование переменной состояния

Приведенная ниже программа заменяет слово magic на first, second, third, в зависимости от того, какая цифра стоит в начале строки.

```
1 int state;  
2  
3 %%  
4 ^1 { state = 1; ECHO; }  
5 ^2 { state = 2; ECHO; }  
6 ^3 { state = 3; ECHO; }  
7 \n { state = 0; ECHO; }  
8 magic { switch (state) {  
9 case 1: printf("<first >"); break;  
10 case 2: printf("<second >"); break;  
11 case 3: printf("<third >"); break;  
12 default : ECHO;  
13 }  
14 }  
15 %%  
16  
17 #include "yy.c"
```

Тестовые данные, подаваемые на вход, приведены ниже:

```
1 magic Cras pulvinar nisi libero , eu pharetra sapien elementum et  
2 1 Fusce id justo tempor, sollicitudin diam non  
3 3 vestibulum lorem Fusce pellentesque magicante at  
4 2 leo mattis magic condimentum.  
5 magic 1  
6 2 magic magic condimentum
```

В результате получили:

```
1 magic Cras pulvinar nisi libero , eu pharetra sapien elementum et
2 1 Fusce id justo tempor, sollicitudin diam non
3 3 vestibulum lorem Fusce pellentesque <third>ante at
4 2 leo mattis <second> condimentum.
5 magic 1
6 2 <second> <second> condimentum
```

Программа отработала корректно.

2.15 Решение предыдущей задачи при помощи стартовых условий

Программа была модифицирована для решения той же самой задачи, но уже с использованием стартовых условий.

```
1 %START c1 c2 c3
2
3 %%
4 ^1      { ECHO; BEGIN c1; }
5 ^2      { ECHO; BEGIN c2; }
6 ^3      { ECHO; BEGIN c3; }
7 \n      { ECHO; BEGIN 0; }
8 <c1>magic printf( "<first>" );
9 <c2>magic printf( "<second>" );
10 <c3>magic printf( "<third>" );
11 %%
12
13 #include "yy.c"
```

Тестовые данные, подаваемые на вход, приведены ниже:

```
1 magic Cras pulvinar nisi libero , eu pharetra sapien elementum et
2 1 Fusce id justo tempor, sollicitudin diam non
3 3 vestibulum lorem Fusce pellentesque magicante at
4 2 leo mattis magic condimentum.
5 magic 1
6 2 magic magic condimentum
```

В результате получили:

```
1 magic Cras pulvinar nisi libero , eu pharetra sapien elementum et
2 1 Fusce id justo tempor, sollicitudin diam non
3 3 vestibulum lorem Fusce pellentesque <third>ante at
4 2 leo mattis <second> condimentum.
5 magic 1
6 2 <second> <second> condimentum
```

Программа отработала корректно. Результат был получен тот же самый.

2.16 Трассировка стартовых условий

Опять решается та же задача с модифицированной программой. Добавлена трассировка стартовых условий.

```

1 %START c1 c2 c3
2
3 %{
4 #define YY_USER_ACTION { fprintf(stderr, "<%d>", YYSTATE); }
5 %}
6
7 %%
8 ^1      { ECHO; BEGIN c1; }
9 ^2      { ECHO; BEGIN c2; }
10 ^3      { ECHO; BEGIN c3; }
11 \n      { ECHO; BEGIN 0; }
12 <c1>magic printf( "<first>" );
13 <c2>magic printf( "<second>" );
14 <c3>magic printf( "<third>" );
15 %%
16
17 #include "yy.c"

```

Тестовые данные, подаваемые на вход, приведены ниже:

```

1 magic Cras pulvinar nisi libero , eu pharetra sapien elementum et
2 1 Fusce id justo tempor, sollicitudin diam non
3 3 vestibulum lorem Fusce pellentesque magicante at
4 2 leo mattis magic condimentum.
5 magic 1
6 2 magic magic condimentum

```

В результате получили:

```

1 magic Cras pulvinar nisi libero , eu pharetra sapien elementum et
2 1 Fusce id justo tempor, sollicitudin diam non
3 3 vestibulum lorem Fusce pellentesque <third>ante at
4 2 leo mattis <second> condimentum.
5 magic 1
6 2 <second> <second> condimentum

```

Программа отработала корректно. Результат был получен тот же самый. Кроме того, программа вывела в консоль следующее сообщение:

```

<0><0><0><0><0><0><0><0><0><0><0><0>
<0><0><0><0><0><0><0><0><0><0><0><0>
<0><0><0><0><0><0><0><0><0><0><0><0>
<0><0><0><0><0><0><0><0><0><0><0><0>
<0><0><0><0><0><0><0><0><0><0><0><0>
<1><1><1><1><1><1><1><1><1><1><1><1>
<1><1><1><1><1><1><1><1><1><1><1><1>
<1><1><1><1><1><1><1><1><1><1><1><1>
<1><1><1><1><1><1><1><1><1><1><1><1>
<1><1><1><1><0><3><3><3><3><3><3><3>
<3><3><3><3><3><3><3><3><3><3><3><3>
<3><3><3><3><3><3><3><3><3><3><3><3>
<3><3><3><3><3><3><3><3><3><3><3><3>
<3><3><3><0><2><2><2><2><2><2><2><2>
<2><2><2><2><2><2><2><2><2><2><2><2>

```

```

<2><2><2><2><2><2><2><0><0><0><0><0>
<0><0><0><0><2><2><2><2><2><2><2><2>
<2><2><2><2><2><2><2><2>

```

2.17 Подсчет количества she и he без учета he внутри she

Приведенная ниже программа вычисляет количество he и she в тексте.

```

1      int s = 0, h = 0;
2
3  %%
4  she      s++;
5  he       h++;
6  .|\n    ;
7  %%
8
9  #ifndef yywrap
10 int yywrap() { return( 1 ); }
11 #endif
12
13 main()
14 {
15     while( yylex() );
16     printf( "she: %d times, he: %d times\n", s, h );
17 }

```

Тестовые данные, подаваемые на вход, приведены ниже:

```

1 Praesent arcu esthe congue sed she she heehe posuere sit amet
2 dictum ac dui he Utshe augue purus consectetur ut
3 porttitor sed mollis eget she turpis Pellentesque elit felis
4 posuere vitae he libero id sollicitudin she porta sapien

```

В результате получили:

```

1 she: 5 times, he: 5 times

```

Программа отработала корректно.

2.18 Подсчет всех экземпляров she и he с использованием REJECT

Приведенная выше программа была модифицирована с использованием команды REJECT.

```

1      int s = 0, h = 0;
2
3  %%
4  she      { s++; REJECT; }
5  he       { h++; REJECT; }
6  .|\n    ;
7  %%
8
9  #ifndef yywrap
10 int yywrap() { return( 1 ); }
11 #endif

```

```

12
13 main()
14 {
15     while( yylex() );
16     printf( "she: %d times, he: %d times\n", s, h );
17 }

```

Тестовые данные, подаваемые на вход, приведены ниже:

```

1 Praesent arcu esthe congue sed she she heehe posuere sit amet
2 dictum ac dui he Utshe augue purus consectetur ut
3 porttitor sed mollis eget she turpis Pellentesque elit felis
4 posuere vitae he libero id sollicitudin she porta sapien

```

В результате получили:

```

1 she: 5 times, he: 5 times

```

Программа отработала корректно. Результат соответствует тому, что мы получили ранее.

2.19 Подсчет she и he с использованием yyless

Приведенная выше программа была модифицирована с заменой REJECT на yyless.

```

1     int s = 0, h = 0;
2 %%
3 she   { s++; yyless(1); }
4 he    { h++; }
5 .|\n  ;
6 %%
7
8 #ifndef yywrap
9 int yywrap() { return 1; }
10 #endif
11
12 main ()
13 {
14     while (yylex());
15     printf("she: %d times, he: %d times\n", s, h);
16 }

```

Тестовые данные, подаваемые на вход, приведены ниже:

```

1 Praesent arcu esthe congue sed she she heehe posuere sit amet
2 dictum ac dui he Utshe augue purus consectetur ut
3 porttitor sed mollis eget she turpis Pellentesque elit felis
4 posuere vitae he libero id sollicitudin she porta sapien

```

В результате получили:

```

1 she: 5 times, he: 5 times

```

Программа отработала корректно. Результат соответствует тому, что мы получили ранее.

3 Вывод

В результате выполнения работы были получены навыки работы с генератором программ лексической обработки текстов Lex. Было изучено и запущено четырнадцать программ, тексты которых были распространены преподавателем. Также была разработана собственная программа, в соответствии с индивидуальным заданием. Программы были транслированы в текста на языке C, скомпилированы и запущены с различными входными данными. Эксперименты, которые наиболее полно отражают суть работы программы, приведены в отчёте. Проведенная работа позволила изучить проблемы, возникающие при разработке и шаблоны, которые могут служить решением для них.