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 Мальцев М.С.
Преподаватель Цыган В.Н.

Санкт-Петербург
23 марта 2019 г.

1 Цель работы

- Познакомиться с языком синтаксического разбора *yacc*
- Выполнить трансляцию предложенных программ на языке *yacc*. Протестировать их работу.

2 Ход работы

2.1 Проверка формата даты

Используемая программа:

```
%token   NUMBER MONTH
%start    date

%%
date :   MONTH NUMBER NUMBER
%%
```

```
%{
#include "y.tab.h"
%}

%%
[0-9]+       { return NUMBER; }
jan          |
feb          |
march        |
apr          |
may          |
june         |
july         |
aug          |
sep          |
oct          |
nov          |
dec          { return MONTH; }
[ \t\n]      ;
.            { return 0; }
%%

#ifndef yywrap
int yywrap () { return 1; }
#endif
```

Сгенерированный файл:

```
#include <stdio.h>

/* ------- Define external objects ------- */

int yydebug = 0;     /* To trace parser, set yydebug = 1      */
             /* ... and call yacc with options -vtd  */
             /* To not trace, set yydebug = 0        */
             /* ... and call yacc with option -d     */
```

```
9
10 /* You can use "yyerror" for your own messages */
11 yyerror (char *s)
12 {
13     fprintf( stderr, "?-%s\n", s );
14 }
15
16 /* ------- Define starting point ------- */
17
18 main ()
19 {
20     return yyparse();
21 }
```

Входные данные:

```
1 jan 12 89!
```

Программа отработала корректно. Так как ошибок обнаружено не было, то вывод программы тоже оказался пустым.

Входные данные были модифицированы:

```
1 jan 12 89 84!
```

В итоге, в консоль было выведено сообщение $? - syntaxerror$. Что сведетельствует, о несоответствии входных данных разработаному шаблону формата даты.

2.2 Изменение структуры ввода даты

Программа приведенная в предыдущем задании была модифицирована:

```
1 %token   NUMBER MONTH
2 %start    date
3
4 %%
5 date :   MONTH NUMBER ',' NUMBER
6 %%
```

```
1 %{
2 #include "y.tab.h"
3 %}
4
5 %%
6 [0-9]+       { return NUMBER; }
7 jan          |
8 feb          |
9 march        |
10 apr         |
11 may         |
12 june        |
13 july        |
14 aug         |
15 sep         |
16 oct         |
17 nov         |
18 dec         { return MONTH; }
```

```
","         { return yytext[0]; }
[ \t\n]     ;
.           { return 0; }
%%

#ifndef yywrap
int yywrap () { return 1; }
#endif
```

Входные данные, которые привели к успешному завершению:

```
dec 12,12
apr 1,1
nov 89,21
```

Входные данные, которые привели к ошибке:

```
dec 12,12,
apr 1.1
```

2.3 Доступ к семантическим значениям

Используемая программа:

```
%token  NUMBER MONTH
%start  date

%%
date :  MONTH NUMBER ',' NUMBER
            { printf("m-d-y: %2u-%2u-%4u\n", $1+1, $2, $4); }
%%
```

```
%{
#include <stdlib.h>
#include "y.tab.h"

#define YYSTYPE int
extern YYSTYPE yylval;
%}

%%
[0-9]+      { yylval = atoi(yytext); return NUMBER; }
jan         { yylval =  0; return MONTH; }
feb         { yylval =  1; return MONTH; }
march       { yylval =  2; return MONTH; }
apr         { yylval =  3; return MONTH; }
may         { yylval =  4; return MONTH; }
june        { yylval =  5; return MONTH; }
july        { yylval =  6; return MONTH; }
aug         { yylval =  7; return MONTH; }
sep         { yylval =  8; return MONTH; }
oct         { yylval =  9; return MONTH; }
nov         { yylval = 10; return MONTH; }
dec         { yylval = 11; return MONTH; }
","         { return yytext[0]; }
[ \t\n]     ;
.           { return 0; }
%%

```

```
#ifndef yywrap
int yywrap () { return 1; }
#endif
```

На вход программы было подано:

```
aug 1,1
```

В итоге получили:

```
m-d-y:  8- 1-    1
```

Следовательно можно сделать вывод, что программа отработала корректно.

2.4 Проверка даты и вывод количества дней от 1970 г.

Используемая программа:

```
%{
long abs_date (int, int, int);  /* month (0-11), day, year */
%}

%token  NUMBER MONTH
%start  date

%%
date :  MONTH NUMBER ',' NUMBER
            { printf("%ld\n", abs_date($1, $2, $4)); }
%%
```

```
%{
#include <stdlib.h>
#include "y.tab.h"

#define YYSTYPE int
extern YYSTYPE yylval;
%}

%%
[0-9]+      { yylval = atoi(yytext); return NUMBER; }
jan         { yylval =  0; return MONTH; }
feb         { yylval =  1; return MONTH; }
march       { yylval =  2; return MONTH; }
apr         { yylval =  3; return MONTH; }
may         { yylval =  4; return MONTH; }
june        { yylval =  5; return MONTH; }
july        { yylval =  6; return MONTH; }
aug         { yylval =  7; return MONTH; }
sep         { yylval =  8; return MONTH; }
oct         { yylval =  9; return MONTH; }
nov         { yylval = 10; return MONTH; }
dec         { yylval = 11; return MONTH; }
","         { return yytext[0]; }
[ \t\n]     ;
.           { return 0; }
%%

#ifndef yywrap
```

```
int yywrap () { return 1; }
#endif
```

На вход программы было подано:

```
jan 1, 1975
```

В итоге получили:

```
1825
```

Следовательно можно сделать вывод, что программа отработала корректно.

2.5 Семантическое значение date и вычисление разницы между датами

Используемая программа:

```
%{
long abs_date (int, int, int);  /* month (0-11), day, year */
%}

%token  NUMBER MONTH
%start  between

%%
date :    MONTH NUMBER ',' NUMBER
            { $$ = abs_date($1, $2, $4); }

between : date '-' date
            { printf("%ld\n", $1 - $3); }
%%
```

```
%{
#include <stdlib.h>
#include "y.tab.h"


extern int yylval;
%}

%%
[0-9]+      { yylval = atoi(yytext); return NUMBER; }
jan         { yylval =  0; return MONTH; }
feb         { yylval =  1; return MONTH; }
march       { yylval =  2; return MONTH; }
apr         { yylval =  3; return MONTH; }
may         { yylval =  4; return MONTH; }
june        { yylval =  5; return MONTH; }
july        { yylval =  6; return MONTH; }
aug         { yylval =  7; return MONTH; }
sep         { yylval =  8; return MONTH; }
oct         { yylval =  9; return MONTH; }
nov         { yylval = 10; return MONTH; }
dec         { yylval = 11; return MONTH; }
[ \t\n]     ;
.           { return yytext[0]; }   /* literal */
```

```
%%

#ifndef yywrap
int yywrap () { return 1; }
#endif
```

На вход программы было подано:

```
feb 29,2000 - dec 31,1999
```

В итоге получили:

```
60
```

Следовательно можно сделать вывод, что программа отработала корректно.

2.6 Определение сопутствующего типа значения нескольких типов

Используемая программа:

```
%{
long abs_date (int m, int d, int y);
%}

%union
{
    int   ival;
    long  lval;
};

%token   <ival> NUMBER MONTH
%type    <lval> date
%start   between

%%
date :    MONTH NUMBER ',' NUMBER
            { $$ = abs_date($1, $2, $4); }

between : date '-' date
            { printf("%ld\n", $1 - $3); }
%%
```

```
%{
#include <stdlib.h>
#include "y.tab.h"
%}

%%
[0-9]+       { yylval.ival = atoi(yytext); return NUMBER; }
jan          { yylval.ival =  0; return MONTH; }
feb          { yylval.ival =  1; return MONTH; }
march        { yylval.ival =  2; return MONTH; }
apr          { yylval.ival =  3; return MONTH; }
may          { yylval.ival =  4; return MONTH; }
june         { yylval.ival =  5; return MONTH; }
july         { yylval.ival =  6; return MONTH; }
```

```
aug         { yylval.ival =  7; return MONTH; }
sep         { yylval.ival =  8; return MONTH; }
oct         { yylval.ival =  9; return MONTH; }
nov         { yylval.ival = 10; return MONTH; }
dec         { yylval.ival = 11; return MONTH; }
[ \t\n]     ;
.           { return yytext[0]; }
%%

#ifndef yywrap
int yywrap () { return 1; }
#endif
```

На вход программы было подано:

```
feb 29,2000 - dec 31,1999
```

В итоге получили:

```
70
```

Следовательно можно сделать вывод, что программа отработала корректно.

2.7 Рекурсивные правила

Используемая программа:

```
%token NUM
%start __list

%%
__list: _list       { printf("No. of items: %d\n", $1); }

_list: /* empty */  { $$ = 0; /* size is 0 */ }
     | list         /* not empty, $$ == $1 by default */
     ;

list: NUM           { $$ = 1; }       /* size := 1 */
    | NUM ',' list  { $$ = $3 + 1; }  /* size := size of sublist + 1 */
    ;
%%
```

```
%{
#include <stdlib.h>
#include "y.tab.h"

#define YYSTYPE int
extern YYSTYPE yylval;    /* value of numeric token */
%}

%%
[0-9]+      { yylval = atoi(yytext); return NUM; }
(.|\n)      return yytext[0];
%%

#ifndef yywrap
int yywrap () { return 1; }
#endif
```

На вход программы было подано:

```
1,2,3,4,5,6,7,8,9,10,20,30,40,50,60,70,80,90,100
```

В итоге получили:

```
No. of items: 19
```

Следовательно можно сделать вывод, что программа отработала корректно.

2.8 Использование рекурсии при чтении списка

Используемая программа:

```
%token NUM
%start __list

%%
__list: _list

_list: /* empty */  { $$ = 0; }
      | list
      ;

list: NUM       { $$ = 1; print($$, $1, 1); }
    | NUM
      ','
      list      { $$ = $3 + 1; print($$, $1, 2); }
    ;
%%

print (int len, int val, int rule)
{
    printf("%d: %d (rule %d)\n", len, val, rule) ;
}
```

```
%{
#include <stdlib.h>
#include "y.tab.h"

#define YYSTYPE int
extern YYSTYPE yylval;     /* value of numeric token */
%}

%%
[0-9]+  { yylval = atoi(yytext); return NUM; }
\n      return '\n';
.       return yytext[0];
%%

#ifndef yywrap
int yywrap () { return 1; }
#endif
```

На вход программы было подано:

```
1,2,3,4,5,6,7,8,9,10,20,30,40,50,60,70,80,90,100
```

В итоге получили:

```
1: 100 (rule 1)
2: 90 (rule 2)
3: 80 (rule 2)
4: 70 (rule 2)
5: 60 (rule 2)
6: 50 (rule 2)
7: 40 (rule 2)
8: 30 (rule 2)
9: 20 (rule 2)
10: 10 (rule 2)
11: 9 (rule 2)
12: 8 (rule 2)
13: 7 (rule 2)
14: 6 (rule 2)
15: 5 (rule 2)
16: 4 (rule 2)
17: 3 (rule 2)
18: 2 (rule 2)
19: 1 (rule 2)
```

Следовательно можно сделать вывод, что программа отработала корректно.

2.9 Дополнительное микрозадание

3 Вывод