



Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ имени Н.Э.БАУМАНА

(национальный исследовательский университет)»

Факультет: Информатика и системы управления

Кафедра: Теоретическая информатика и компьютерные технологии

Лабораторная работа № 6

Раскрутка самоприменимого компилятора

Вариант 28

Работу выполнил

студент группы ИУ9-61

Бакланова А.Д.

Москва, 2020

Цель работы:

Целью данной работы является изучение генератора лексических анализаторов flex.

Индивидуальный вариант:

28	Целочисленные константы: последовательности десятичных цифр, предваряемые символом #. Имена переменных: последовательности десятичных цифр, начинающиеся со знаков «.» или «:», имена массивов: последовательности десятичных цифр, начинающиеся с «,» или «;». Ключевые слова «PLEASE», «DO», «FORGET».
----	--

Задание:

Требуется воспользоваться генератором лексических анализаторов flex для порождения лексического анализатора на языке C (или C++).

Порождённый лексический анализатор должен загружать входной поток из файла (кодировка – ASCII) и выводить в стандартный поток вывода описания распознанных лексем в формате

Тег (координаты_фрагмента): атрибут лексемы

При этом для лексем, не имеющих атрибутов, нужно выводить только тег и координаты. Например,

```
IDENT (1, 2)-(1, 4): str
ASSIGN (1, 8)-(1, 9):
STRING (1, 11)-(1, 16): qwerty
```

Лексемы во входном файле могут разделяться пробельными символами (пробел, горизонтальная табуляция, маркеры окончания строки), а могут быть записаны слитно (если это не приводит к противоречиям).

Идентификаторы и числовые литералы не могут содержать внутри себя пробельных символов, если в задании явно не указано иного (варианты 4, 14 и 36). Комментарии, строковые и символьные литералы могут содержать внутри себя пробельные символы.

Входной файл может содержать ошибки, при обнаружении которых лексический анализатор должен выдавать сообщение с указанием координаты, восстанавливаться и продолжать работу.

Варианты языков для лексического анализа приведены в таблицах 1, 2, 3, 4 и 5.

Реализация:

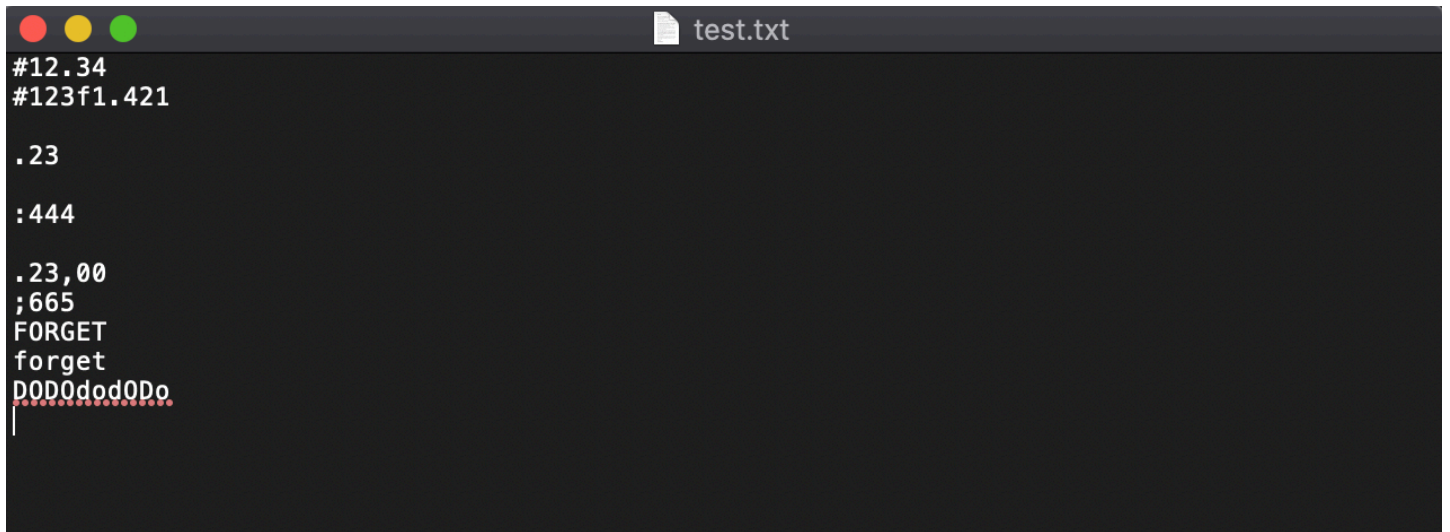
```
1 %option noyywrap bison-bridge bison-locations
2
3 %{
4
5 #include <stdio.h>
6 #include <stdlib.h>
7
8 #define ARRAY 1
9 #define WORD 2
10 #define TAG_VARIABLE 3
11 #define TAG_WORD 4
12 #define TAG_SPACE 5
13
14
15 const char *tag_names [] = {
16     "END_OF_PROGRAM", "ARRAY", "WORD", "VARIABLE", "CONSTANT", "SPACE"
17 };
18
19 struct Position {
20     int line, pos, index;
21 };
22
23 void print_pos(struct Position *p) {
24     printf("(%d,%d)", p->line, p->pos);
25 }
26
27 struct Fragment {
28     struct Position starting, following;
29 };
30
31 typedef struct Fragment YYLTYPE;
32
33 void print_frag(struct Fragment *f) {
34     print_pos(&(f->starting));
35     printf("-");
36     print_pos(&(f->following));
37 }
38
39 union Token {
40     const char *ident;
41     const char *word;
42 };
43
44 typedef union Token YYSTYPE;
45
46 int continued;
47 struct Position cur;
48
49 #define YY_USER_ACTION
50 {
51     int i;
52     if (!continued)
53         yylloc->starting = cur;
54     continued = 0;
55
56     for (i = 0; i < yyleng; i++) {
57         if (yytext[i] == '\n') {
58             cur.line++;
59             cur.pos = 1;
60         } else {
61             cur.pos++;
62         }
63         cur.index++;
64     }
65     yylloc->following = cur;
66 }
```

```

68 void init_scanner(const char *path) {
69     continued = 0;
70     cur.line = 1;
71     cur.pos = 1;
72     cur.index = 0;
73     yyin = fopen(path, "r");
74 }
75
76 void err(const char *msg) {
77     printf("Error");
78     print_pos(&cur);
79     printf(": %s\n", msg);
80 }
81
82 %%
83
84 SPACE [ \t\r\n\v\f]+
85 DIGIT [0-9]
86
87 CONSTANT #{DIGIT}*
88 VARIABLE (\.|\:){DIGIT}*
89 ARRAY (\\,|\\;){DIGIT}*
90 WORD (PLAESE|DO|FORGET)
91
92 %%
93 {SPACE} {
94     yylval->ident = yytext;
95     BEGIN(0);
96     return TAG_SPACE;
97 }
98 {CONSTANT} {
99     yylval->ident = yytext;
100     BEGIN(0);
101     return TAG_WORD;
102 }
103 {ARRAY} {
104     yylval->ident = yytext;
105     BEGIN(0);
106     return ARRAY;
107 }
108 {WORD} {
109     yylval->ident = yytext;
110     BEGIN(0);
111     return WORD;
112 }
113 {VARIABLE} {
114     yylval->ident = yytext;
115     BEGIN(0);
116     return TAG_VARIABLE;
117 }
118
119 . err("error");
120 %%
121

```

Тестирование:



```
#12.34
#123f1.421

.23

:444

.23,00
;665
FORGET
forget
DODOdodODo
```

Вывод:

В результате выполнения лабораторной работы были получены знания о генераторе лексических анализаторов flex.