Name - Vasu Kalariya
Roll - PE29
Sub - SSC

Lab Assignment -5

Aim: Generate lexical analyzer for Java/C language using LEX.

Objective:

1) To understand the lexical analysis phase of the compiler
2) To understand the scanner for subsets of Java

Theory:

→ Token, Lexeme and Pattern

(i) Lexemes are the smallest logical Unit (word) of program of { I, sum, buffer, 10, +, for, ... }

(ii) Token is a set of similar lexemes.

of Identifier - { I, sum, buffer ... }

Keyword - { for, ... }

(iii) Pattern is a regular expression of Digit [0-9]

→ Use of regular expression (RE) in specifying lexical structure of a language

The lexical analyzer needs to scan and identify only a finite set of valid string / token / lexeme that belong to language in head. It searches for pattern defines by the language rules. These patterns are denoted by Regular expressions.

→ Explain format of lex specification file (*.I)

A lex program is seperated into three section by %% delimeters. The format of lex source is as follow:

{ ~~definitions~~
{ definition }
%%
{ rules }
%%
{ user subroutines }

- Definitions: include ~~de~~ declaration of constant, variable and regular definitions.
- Rules define the statement of form p1 { action 1 } p2 { action 2 } etc. where P1 describes the regular expression and action 1 describes the actions the lexical analyzer should taken when pattern pi matches a lexeme
- User subroutines are auxiliary procedure needed by the actions. The subroutine can be loaded with the lexical analyzer and compiled ~~se~~ seperately.

INPUT: Subset of Java language

Output: Sequence of tokens generated by lexical analyzer & symbol table

PLATFORM: Linux (JAVA)

CONCLUSION: Implemented scanner in Java

FAQs

1 Give various tasks performed during lexical analysis phase

(i) Recognizing basic elements
(ii) Removal of white spaces and comments
(iii) Recognizing constants and literals
(iv) Recognizing keywords and identifiers

2 What is the role of RE, DFA in lexical analysis.

The collections of tokens of a programming language can be specified by a set of regular expression. Lexical analyzer for the language use a DFA in its core. Different final states of the DFA identifies different tokens. Synthesis of this DFA form of RE can be automated.

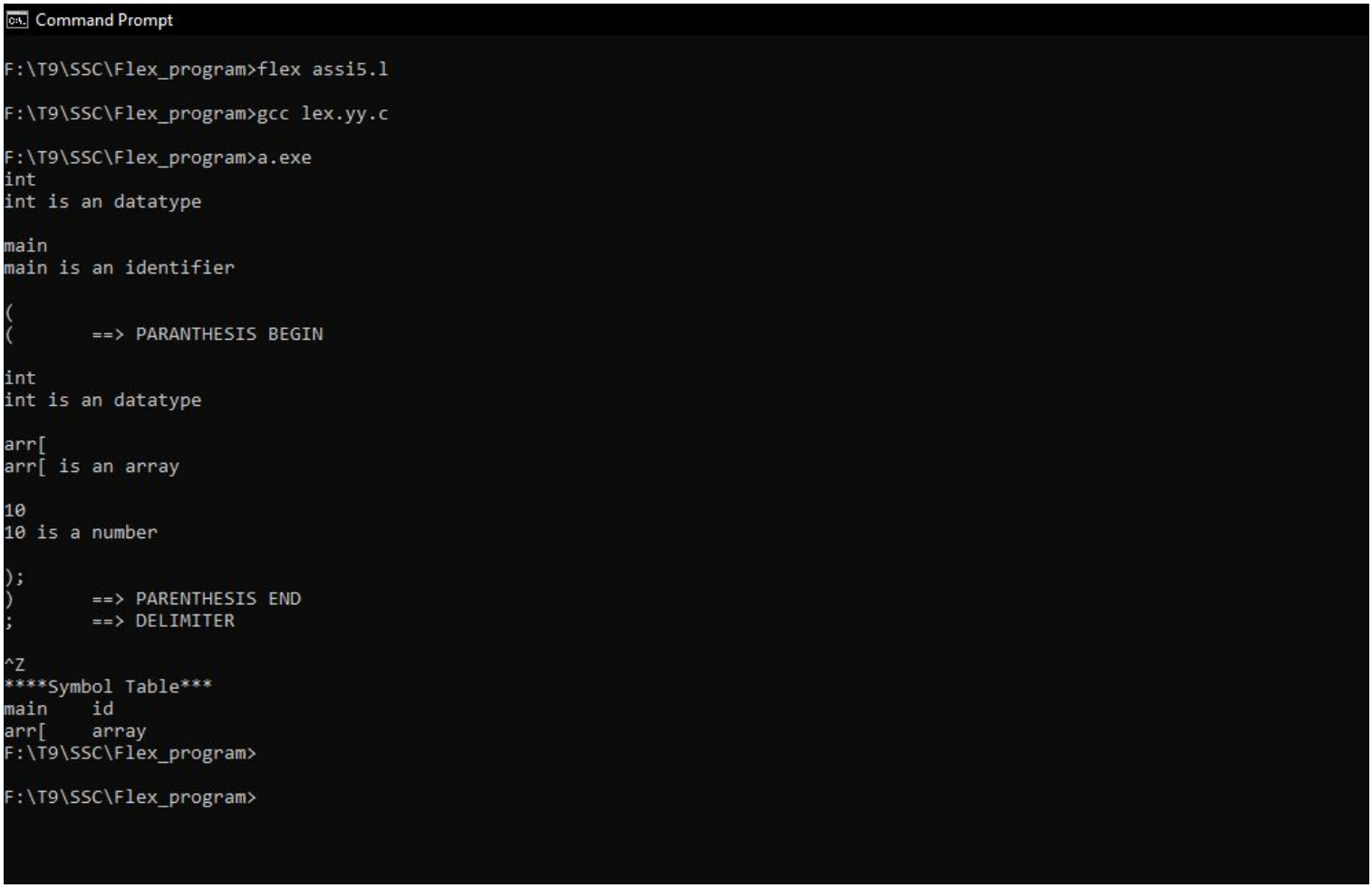
3 What is LEX?

LEX (Lexical analyzer generator) is a program designed to generate scanners, also known as tokenizers, which recognize lexical patterns in text.

```
%option noyywrap
%{
#include<stdio.h>
#include<string.h>
struct SymbolTable
{
char symbol[10];
char type[10];
}SymbolTable[10];
int count = 0;
char data[10];
char type[10];
void insert();
void display();
%}
letter[a-zA-Z]
digit[0-9]
num({digit}{digit}*)
KEYWORDS "class"|"static"
datatype(int|char|float|void)
CONDITIONAL "if"|"else"|"else if"|"switch"|"case"
SC ";"
array({id}(\[))
id({letter}({letter}|{digit})*)
ARITH_OP "+"|"-"|"/"|"%"|"*";
LOGICAL_OP "&&"|"||"|"!"|"!="
REL_OP "<"|">"|"<="|">="|"=="
UNARY "++"|"--"
%%

{datatype} {printf("%s is an datatype\n",yytext);}
{CONDITIONAL} { printf("%s\t==> CONDITIONAL\n",yytext);}
{num} {printf("%s is a number\n",yytext);}
{array} {printf("%s is an array\n",yytext);insert(yytext,"array");}
{KEYWORDS} {printf("%s\t==> KEYWORDS\n",yytext);}
{id} {printf("%s is an identifier\n",yytext);insert(yytext,"id");}
{UNARY} {printf("%s\t==> UNARY OP\n",yytext);}
{ARITH_OP} {printf("%s\t==> ARITHMETIC OPERATOR\n",yytext);}
{LOGICAL_OP} {printf("%s\t==> LOGICAL OP\n",yytext);}
{SC} {printf("%s\t==> DELIMITER\n",yytext);}
"=" {printf("%s\t==> ASSIGNMENT OP\n",yytext);}
"{" {printf("%s\t==> BLOCK BEGIN\n",yytext);}
"}" {printf("%s\t==> BLOCK END\n",yytext);}
"(" {printf("%s\t==> PARANTHESIS BEGIN\n",yytext);}
")" {printf("%s\t==> PARENTHESIS END\n",yytext);}
%%
int main()
{
yylex();
display();
```

```
return 0;
}
void insert(char data[10],char type[10])
{strcpy(SymbolTable[count].symbol,data);
strcpy(SymbolTable[count].type,type);
++count;
}
void display()
{
int i;
printf("****Symbol Table***");
for(int i=0;i<count;i++)
{
printf("\n%s\t%s",SymbolTable[i].symbol,SymbolTable[i].type);
}
}
```


Command Prompt
F:\T9\SSC\Flex_program>flex assi5.l
F:\T9\SSC\Flex_program>gcc lex.yy.c
F:\T9\SSC\Flex_program>a.exe
int
int is an datatype
main
main is an identifier
(
(==> PARANTHESIS BEGIN
int
int is an datatype
arr[
arr[is an array
10
10 is a number
);
)   ==> PARENTHESIS END
;   ==> DELIMITER
^Z
****Symbol Table***
main   id
arr[array
F:\T9\SSC\Flex_program>
F:\T9\SSC\Flex_program>