UCS 1602 - Compiler Design

Assignment – 2 -Lexical Analyser Using Lex Tool

Swetha Saseendran
CSE-C
185001183

Aim:

To develop a Lexical analyzer to recognize the patterns namely, identifiers, constants, comments and operators using the following regular expressions and construct symbol table for the identifiers with the following information.

Code:

```
/*lex program for lexical analyzer*/
%{
#include<stdio.h>
#include<string.h>

typedef struct
{
    char var[10];
    char type[10];
    char value[10];
}symbol_table;

symbol_table t[10];
int ind=0;
char type[10];

int insert(char *token)
{
    int i;
    for(i=0;i<=ind;i++)
    {
        if(strcmp(token,t[i].var)==0)
            return 0;
    }
    return 1;
```

```
}
%}

keyword int|float|double|char|do|while|for|if|break|continue|void|return|else|
string
function main|printf|scanf|getchar|getch
strconst \".*\"
preprocessor #.+
identifier [_a-zA-Z][a-zA-Z0-9]*
numconst [0-9]+|[0-9]+[.][0-9]+
special_char [{}(),;]
sing_comment [//].*
multi_comment "/*"(.|\n)*"*/"
relational "<="|">="|"<"|"=="|">"
arithmetic "+"|"-"|"--"|"++"|"%"|"*"|"/"
assign "="|"+="|"-="|"/="|"%="
logical "||"|"&&"
bitwise "<<"|">>"|"^"|"~"

%%

{keyword} {
    printf("KEYWORD ");
    if(strcmp(yytext,"int")==0)
    {
        strcpy(type,yytext);
    }
    else if(strcmp(yytext,"float")==0)
    {
        strcpy(type,yytext);
    }
    else if(strcmp(yytext,"char")==0)
    {
        strcpy(type,yytext);
    }
    else if(strcmp(yytext,"double")==0)
    {
        strcpy(type,yytext);
    }
}
{function} printf("FUNCTION\t");
{sing_comment} printf("SINGLE-LINED COMMENT\t");
{multi_comment} printf("MULTI-LINED COMMENT\t");
{logical} printf("LOP\t");
{bitwise} printf("BOP\t");
{numconst} {
    printf("NUMCONST ");
    if(strcmp(t[ind].value,"null")==0)
```

```
    {
        strcpy(t[ind].value,yytext);
        ind++;
    }
}
{strconst} {printf("STRCONST ");}
{preprocessor} printf("PPDIR\t");
{identifier} {
    printf("ID\t");
    if(insert(yytext))
    {
        strcpy(t[ind].type,type);
        strcpy(t[ind].var,yytext);
        strcpy(t[ind].value,"null");
        ind++;
    }
}
{special_char} printf("SP\t");
{relational} printf("RELOP\t");
{arithmetic} printf("AOP\t");
{assign} {
    printf("ASSIGN ");
    ind--;
    }

%%

int yywrap(void){
    return 1;
}

void main()
{
    yyin=fopen("in.txt","r");
    yylex();
    int i;
    printf("\n\nType\tVariable\tValue\n");
    for(i=0;i<=ind;i++)
    {
        printf("%s\t%s\t%s\n",t[i].type,t[i].var,t[i].value);
    }
}
```

INPUT TEXT FILE TO PARSE (in.txt):

C: > Users > Admin > OneDrive > Desktop > Semester VI > Labs and Mini Project > CD Lab > Assignment 2 > in.txt

```
#include<stdio.h>
void main()
{
    int a=10,b=20;
    float c=10.4,d=20.5;
    float ans;
    if(a == b)
        printf("a is greater");
    else
        printf("b is greater");
    for(int i=0;i<10;i++)
    {
        if(c && d)
            ans = c<<2;
        else if(c || d)
            ans = ~d;
        else
            break;
    }
    /*
        Muliti-lined Comment
    */
    //Single-lined Comment
    return 0;
}
```

OUTPUT:

```
Admin@DESKTOP-1883PSF MINGW64 ~/OneDrive/Desktop/Semester VI/Labs and Mini Project/CD Lab/Assignment 2
$ ./a
PPDIR
KEYWORD  FUNCTION       SP      SP
SP
        KEYWORD  ID     ASSIGN NUMCONST SP      ID      ASSIGN NUMCONST SP
        KEYWORD  ID     ASSIGN NUMCONST SP      ID      ASSIGN NUMCONST SP
        KEYWORD  ID     SP
        KEYWORD SP      ID       RELOP   ID     SP
                FUNCTION        SP      STRCONST SP     SP
        KEYWORD
                FUNCTION        SP      STRCONST SP     SP
        KEYWORD SP      KEYWORD  ID     ASSIGN NUMCONST SP      ID      RELOP   NUMCONST SP     ID      AOP     SP
        SP
                KEYWORD SP      ID       LOP     ID     SP
                        ID       ASSIGN  ID     BOP     NUMCONST SP
                KEYWORD  KEYWORD SP     ID       LOP     ID     SP
                        ID       ASSIGN  BOP    ID      SP
                KEYWORD
                        KEYWORD SP
        SP
        MULTI-LINED COMMENT
        SINGLE-LINED COMMENT
        KEYWORD  NUMCONST SP
SP

Type    Variable        Value
int     a       10
int     b       20
float   c       10.4
float   d       20.5
float   ans     0
int     i       0
```

LEARNING OUTCOME:

- Understood the basic functionalities and working of a Lexical Analyser, that it breaks the syntaxes into a series of tokens and analyses it.
- Understood the basic working of the lex tool and that its more powerful and convenient to use for Lexical Analysis task compared to conventional C programming.
- Learnt how to implement regular expressions in lex tool and develop parsers to identify tokens and thereby converting regex to finite automata.
- Understood the working of the symbol table and to implement a basic symbol table using Lex on the parsed C program.
- Also got a good visualization behind the process of compilation and role of Lexical Analyser in it. Also, the assignment helped me to understand how an analyser maps various tokens based on given specifications.

RESULT:

Successfully implemented the code to stimulate Lexical Analyser to scan the entire source code and identify the tokens and form the symbol table for the same using the lex tool.