Exp No: 2

IMPLEMENTATION OF LEXICAL ANALYSER AND SYMBOL TABLE IN LEX TOOL

Date: 09/02/2021

Name: MADHUMITHA S

Register Number: 185001086

---

1) **Aim:**

To implement Lexical Analyser and Symbol Table in Lex tool to identify and tokenise identifiers, constants, comments etc.

2)PROGRAM CODE:

```
%{
      #include<stdio.h>
      #include<string.h>
     int flag=0,f1=0,f2=0;
     char val[10][2];
     char ids[10][15];
     int typ[10];
     int pos1=0,pos2=0,pos3=0;
%}
multicomm  ^\/\*
multicommend (\*\/)$
key
auto|break|case|char|const|continue|default|do|double|else|enum|extern|floa
t|for|goto|if|int|long|register|return|short|signed|sizeof|static|struct|sw
itch|typedef|union|unsigned|void|volatile|while
func      (printf|scanf|getch|clrscr|main)\(.*\)
identifier [A-Za-z][A-Za-z0-9]*
num       [0-9][0-9]*
ft        [0-9][0-9]*\.[0-9][0-9]*
arith     [\+\-%\*]
arithssign [\+\-%\*]\=
relational [\<\>!]\=*|\=\=
assign     \=
```

```
logical    &&|\|\||!
bitwise    \<\<|\>\>|&|\^
unary      \+\+|--|-
special    [\[\]\{\}\(\),;\.]
prepro     ^#.*
comm      ^\\\\.*
newlne     \n
stringy   ^".*"$
%%
{multicomm} {flag=1;printf("%s-Multi-Line comment\n",yytext);}
{num}  {if(flag==0){printf("%s-Integer
Constant\n",yytext);if(f1){strcpy(val[pos2++],yytext);typ[pos3++]=2;}}}
{ft} {if(flag==0){printf("%s-
Float\n",yytext);if(f2){strcpy(val[pos2++],yytext);typ[pos3++]=4;}}}
{comm}  {if(flag==0)printf("%s-Single Line comment\n",yytext);}
{stringy}  {if(flag==0)printf("%s-String constant\n",yytext);else
printf("%s-Comment",yytext);}
{key} {if(flag==0){printf("%s-
Keyword\n",yytext);if(strcmp(yytext,"int")==0){f1=1;}else
f1=0;if(strcmp(yytext,"float")==0){f2=1;}else f2=0;}else printf("%s-
Comment\n",yytext);}
{func} {if(flag==0)printf("%s-Function call\n",yytext);}
{identifier} {if(flag==0){printf("%s-
Identifier\n",yytext);strcpy(ids[pos1++],yytext); }else printf("%s-
Comment\n",yytext);}
{arith}  {if(flag==0)printf("%s-Arithmetic operator\n",yytext);}
{arithssign} {if(flag==0)printf("%s-Arithmetic Assignment
operator\n",yytext);}
{relational} {if(flag==0)printf("%s-Relational operator\n",yytext);}
{logical} {if(flag==0)printf("%s-Logical operator\n",yytext);}
{bitwise} {if(flag==0)printf("%s-Bitwise operator\n",yytext);}
{unary} {if(flag==0)printf("%s-Unary operator\n",yytext);}
{special} {if(flag==0)printf("%s-Special character\n",yytext);}
{prepro} {if(flag==0)printf("%s-Pre processor statement\n",yytext);}
{assign}  {if(flag==0)printf("%s-Assignment operator\n",yytext);}

{multicommend} {flag=0;printf("%s-End of multi-line comment\n",yytext);}
```

```
%%
main(){
      FILE *yyin;
      int i=0,addr=1000;
      yyin=fopen("code_02.txt","r");
      yyset_in(yyin);
    yylex();
    printf("\nSymbol table\n");
    printf("Name\tValue\tSize\tAddress\n");
    printf("----\t-----\t----\t-------\n");
    for(i=0;i<pos2;i++){
printf("\n%s\t%s\t%d\t%d-%d\n",ids[i],val[i],typ[i],addr,addr+typ[i]);
addr=addr+typ[i];
    }
    return 0;

}
```

3)OUTPUT SCREENSHOTS:

Sample code fed as input:

```
#include<stdio.h>
#include<string.h>
main(){
        int a=10,b=10;
        if(a>b){
        printf("a is greater");
        }
        else if(b>a)
        printf("b is greater");
        else
        printf("Both are equal");
        return 0;
}
```

```
madhu@madhu-HP-Pavilion-x360-Convertible-14-dh1xxx:~/cd_lab$ lex A02.l
madhu@madhu-HP-Pavilion-x360-Convertible-14-dh1xxx:~/cd_lab$ gcc lex.yy.c -ll
A02.l:50:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   50 | main(){
      | ^~~~
madhu@madhu-HP-Pavilion-x360-Convertible-14-dh1xxx:~/cd_lab$ ./a.out A02.l
#include<stdio.h>-Pre processor statement

#include<string.h>-Pre processor statement

main()-Function call
{-Special character

        int-Keyword
 a-Identifier
=-Assignment operator
10-Integer Constant
,-Special character
b-Identifier
=-Assignment operator
10-Integer Constant
;-Special character

        if-Keyword
(-Special character
a-Identifier
>-Relational operator
b-Identifier
)-Special character
{-Special character
```

```
madhu@madhu-HP-Pavilion-x360-Convertible-14-dh1xxx: ~/cd_lab
        printf("a is greater")-Function call
;-Special character

        }-Special character

        else-Keyword
 if-Keyword
(-Special character
b-Identifier
>-Relational operator
a-Identifier
)-Special character

        printf("b is greater")-Function call
;-Special character

        else-Keyword

        printf("Both are equal")-Function call
;-Special character

        return-Keyword
 0-Integer Constant
;-Special character

}-Special character

Symbol table
Name    Value   Size    Address
----    -----   ----    -------

a       1010    2       1000-1002

b       10      2       1002-1004
```

4) LEARNING OUTCOME:

- Implementing a lexical analyser that separates the given code into tokens and each token is parsed about and analysed.

- Maintenance of a symbol table in the compiler during the lexical analysis phase storing address and values of identifiers.
- Learnt briefly about the lexical analysis phase of a compiler.
- Using LEX tool for the above.