SSN COLLEGE OF ENGINEERING

ASSIGNMENT 2: LEXICAL ANALYSER USING LEX TOOL

-SRINITHYEE S K

185001166

Code:

```
%{
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

struct symbol{
    char type[10];
    char name[20];
    char value[100];
};  //For Symbol Table

typedef struct symbol sym;

sym sym_table[1000];
int cur_size = -1;
char current_type[10];
%}

number_const [-+]?[0-9]+(\.[0-9]+)?
char_const \'.\'
string_const \".*\"
identifier [a-zA-Z_][a-zA-Z0-9_]*
function [a-zA-Z_][a-zA-Z0-9]*[(].*[)]
keyword
(int|float|char|unsigned|typedef|struct|return|continue|break|if|else|for|while|do|e
xtern|auto|case|switch|enum|goto|long|double|sizeof|void|default|register)
pp_dir ^[#].*[>]$
rel_ops ("<"|">"|"<="|">="|"=="|"!=")
assign_ops ("="|"+="|"-="|"%="|"/="|"*=")
arith_ops ("+"|"-"|"%"|"/"|"*")
single_cmt [/][/].*
multi_cmt ([/][/].*)|([/][*](.|[\n\r])*[*][/])
spl_chars [{}(),;\[\]]

/*Rules*/

%%

{pp_dir} {
    printf("PPDIR ");
    strcpy(current_type, "INVALID");
}

{keyword} {
    printf("KW ");

    if(strcmp(yytext, "int") == 0){
```

```
        strcpy(current_type, "int");
    }
    else if(strcmp(yytext, "float") == 0){
        strcpy(current_type, "float");
    }
    else if(strcmp(yytext, "double") == 0){
        strcpy(current_type, "double");
    }
    else if(strcmp(yytext, "char") == 0){
        strcpy(current_type, "char");
    }
    else{
        strcpy(current_type, "INVALID");
    }
}

{function} {
    printf("FUNCT ");
}

{identifier} {
    printf("ID ");

    if(strcmp(current_type, "INVALID") != 0){
        cur_size++;
        strcpy(sym_table[cur_size].name, yytext);
        strcpy(sym_table[cur_size].type, current_type);

        if(strcmp(current_type, "char") == 0){
            strcpy(sym_table[cur_size].value, "NULL");
        }
        else if(strcmp(current_type, "int") == 0){
            strcpy(sym_table[cur_size].value, "0");
        }
        else{
            strcpy(sym_table[cur_size].value, "0.0");
        }
    }
}

{single_cmt} {
    printf("SCMT ");
}

{multi_cmt} {
    printf("MCMT ");
}

{number_const} {
    printf("NUM_CONST ");

    if(strcmp(current_type, "INVALID") != 0){
```

```
        strcpy(sym_table[cur_size].value, yytext);
    }
}

{char_const} {
    printf("CHAR_CONST ");

    if(strcmp(current_type, "char") == 0){
        strcpy(sym_table[cur_size].value, yytext);
    }
}

{string_const} {
    printf("STR_CONST ");
}

{rel_ops} {
    printf("REL_OP ");
}

{arith_ops} {
    printf("ARITH_OP ");
}

{assign_ops} {
    printf("ASSIGN_OP ");
}

{spl_chars} {
    if(strcmp(yytext, ";") == 0){
        strcpy(current_type, "INVALID");
    }

 }

\n {
    printf("\n");
}

[ \t] { }


%%

int yywrap(void){
    return 1;
}



int main(int argc, char *argv[]){
    int i = 0;
```

```
    yyin = fopen(argv[1], "r");
    yylex();

    printf("\n\t----------------------------------------\n");

    printf("\n\t\t\tSYMBOL TABLE");
    printf("\n\t\tNAME\tTYPE\tVALUE\n");
    for(i = 0; i <= cur_size; i++){
        printf("\t\t%s\t%s\t%s\n", sym_table[i].name, sym_table[i].type,
sym_table[i].value);
    }

    printf("\t----------------------------------------\n");

    return 0;
}
```

OUTPUT:

```
KW FUNCT
KW ID ASSIGN_OP NUM_CONST ID
KW ID ASSIGN_OP NUM_CONST
KW ID ID ASSIGN_OP CHAR_CONST
KW ID ASSIGN_OP NUM_CONST

FUNCT

ID ASSIGN_OP ID ARITH_OP NUM_CONST

KW ID REL_OP NUM_CONST
FUNCT


KW ID REL_OP NUM_CONST
FUNCT
ID ASSIGN_OP NUM_CONST



SCMT
MCMT

KW NUM_CONST


        ----------------------------------------

                        SYMBOL TABLE
                NAME    TYPE    VALUE
                a       int     1
                b       int     0
                c       int     2
                d       char    NULL
                e       char    'Z'
                f       float   1.23
        ----------------------------------------
```