Department of CSE
SSN College of Engineering

Vishakan Subramanian - 18 5001 196 - Semester VI

12 February 2021

UCS 1602 - Compiler Design

Exercise 2: Lexical Analyser Using Lex Tool

Aim:

To write a program using Lex to perform the basic functionalities of a **Lexical Analyser**, and to form a symbol table on the parsed program.

Code:

```
/* Lexical Analyser Using Lex Tool */

/*Definitions*/

%{
#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|extern|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;
```

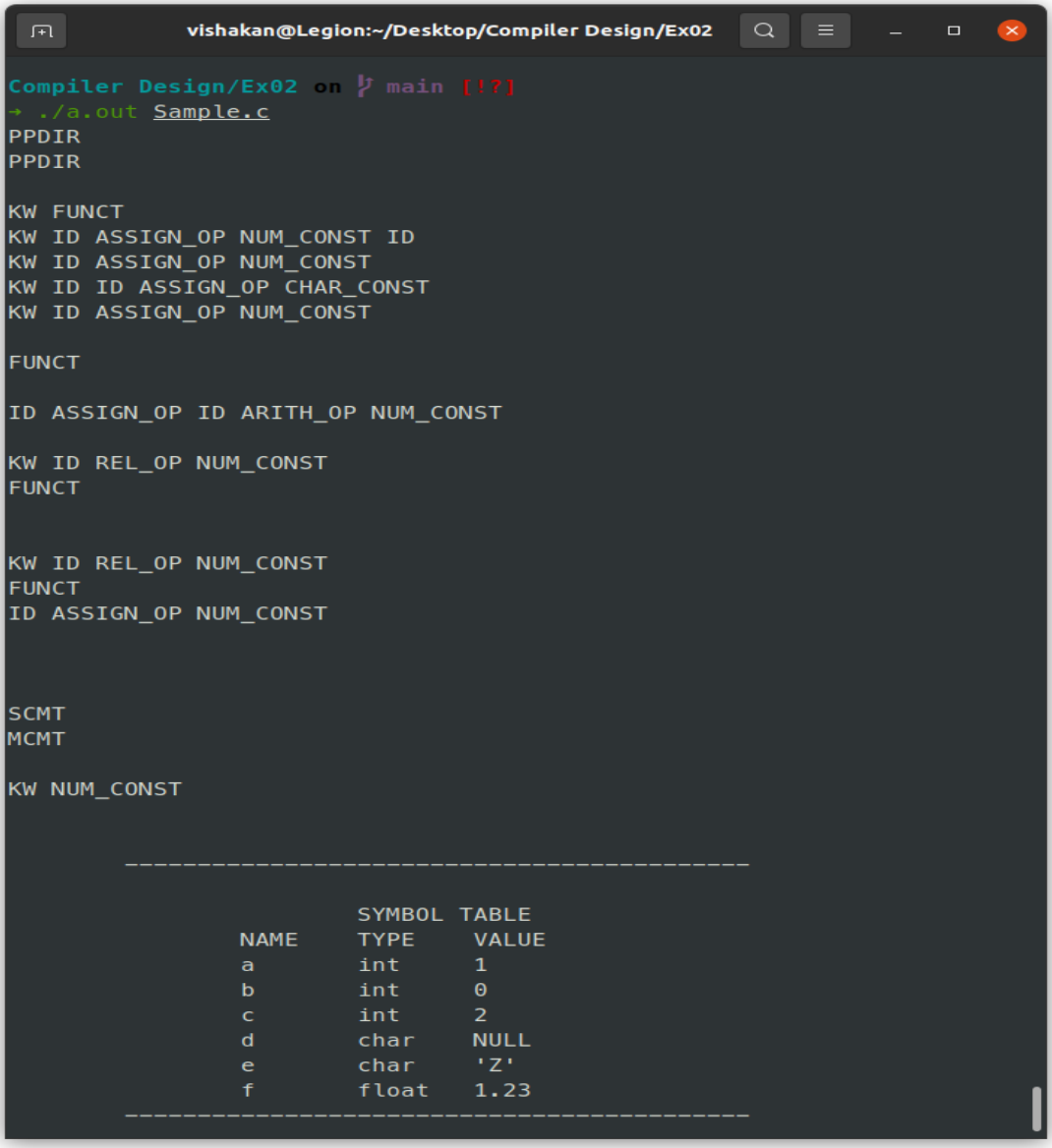
```c
}


/*User Subroutines*/

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;
}
```

Parsed C Code:

```c
#include<stdio.h>
#include<stdlib.h>

int main(){
    int a = 1, b;
    int c = 2;
    char d, e = 'Z';
    float f = 1.23;

    printf("Hello to %d", c);

    a = b + 100;

    if (c > 100){
        printf("Greater");
    }

    while (c > 0) {
        printf("Hello to Lex!");
        c -= 1;
    }


    //a is GREATER than b!
    /* Multi-line
    comment */

    return 0;
}
```

Output:

Figure 1: Console Output



```
vishakan@Legion:~/Desktop/Compiler Design/Ex02

Compiler Design/Ex02 on main [!?]
→ ./a.out Sample.c
PPDIR
PPDIR

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
        ----------------------------------------
```

Learning Outcome:

- From the experiment, I understood the basics of Lex tool.
- I was able to implement recognition for regular expressions using Lex terminology.
- I understood the working of a Lex program.
- I learnt about the three sections of a Lex program, namely, definitions, rules and user subroutines.
- I learnt to implement a basic symbol table using Lex on the parsed C program.
- I understood that Lex tool is more powerful and easy-to-use for Lexical Analysis task compared to conventional C programming.