Department of Computer Science and Engineering

S.G.Shivanirudh , 185001146, Semester VI

1 February 2021

UCS1602 - Compiler Design

Exercise 2: Implementation of Lexical Analyzer

Objective:

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

Code:

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

```

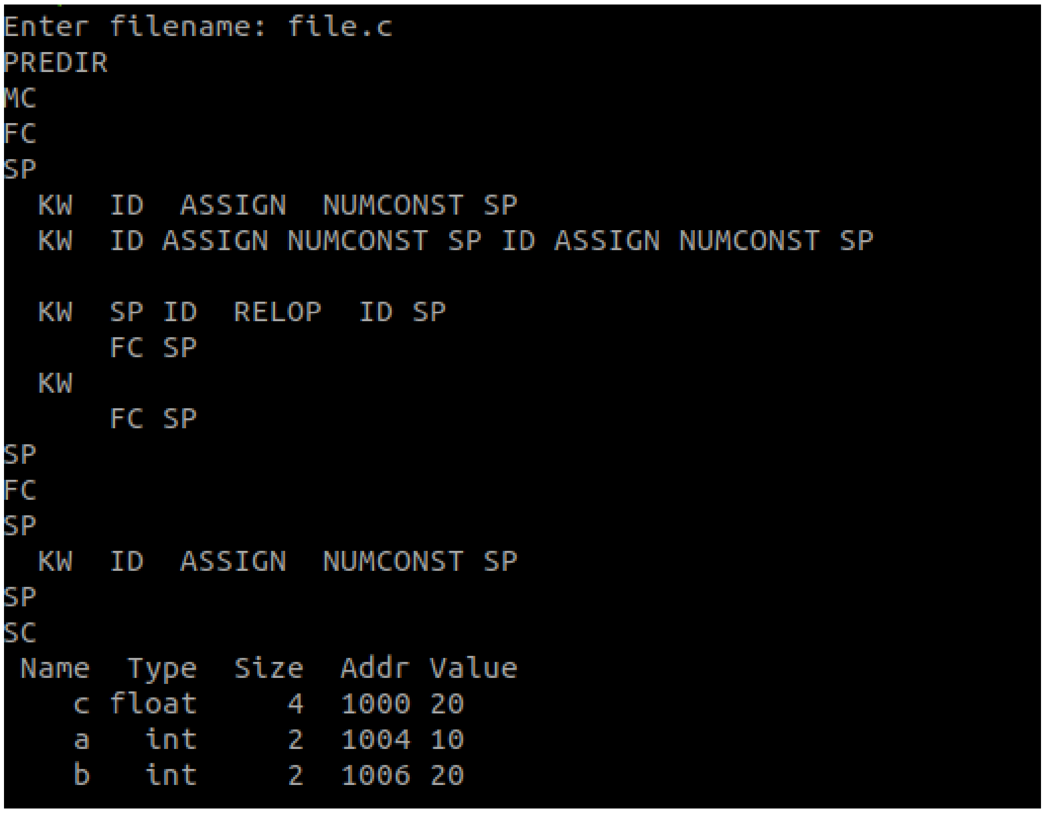
```c
    int symbol_count  = 0, flag=0, fg[20],base = 1000;
    char *symbol_table[100];
    char *values[100];

void set_const(char *val){
    strcpy(val, yytext);
}
void set_flag(int *flag){
    if(strcmp(yytext, "int") == 0)
        *flag = 1;
    else if(strcmp(yytext, "float") == 0)
        *flag = 2;
    else if(strcmp(yytext, "double") == 0)
        *flag = 3;
    else if(strcmp(yytext, "char") == 0)
        *flag = 4;
}

void construct_table(char *symbol_table[], int *symbol_count){
    int size = 0;
    int addr = 1000;
    symbol_table[*symbol_count] = (char*)calloc(100, sizeof(char));
    strcat(symbol_table[*symbol_count], yytext);strcat(symbol_table[*symbol_count], " ");
    if(flag == 1){
        strcat(symbol_table[*symbol_count], "int");strcat(symbol_table[*symbol_count], " ");
        size = 2;
    }
    else if(flag == 2){
        strcat(symbol_table[*symbol_count], "float");strcat(symbol_table[*symbol_count], " ");
        size = 4;
    }
    else if(flag == 3){
        strcat(symbol_table[*symbol_count], "double");strcat(symbol_table[*symbol_count], " ");
        size = 8;
    }
    else if(flag == 4){
        strcat(symbol_table[*symbol_count], "char");strcat(symbol_table[*symbol_count], " ");
        size = 1;
    }
    char *dummy=(char*)calloc(100, sizeof (char));
    sprintf(dummy, "%d", size);
    strcat(symbol_table[*symbol_count], dummy);strcat(
   symbol_table[*symbol_count], " ");
    sprintf(dummy, "%d", base_addr);base_addr += size;
    strcat(symbol_table[*symbol_count], dummy);strcat(
   symbol_table[*symbol_count], " ");
    strcat(symbol_table[*symbol_count], val);strcat(
   symbol_table[*symbol_count], " ");
}
%}
/*Rules*/

/*Preprocessor directives*/
inc #(.)*


/*Keywords*/
kw int|char|float|double|if|else|for|while|do

/*Function*/
funcCall [a-zA-Z]([a-zA-Z]|[0-9])*\(

/*ID*/
id [a-zA-Z]([a-zA-Z]|[0-9])*

/*Constant*/

numConst [0-9]+
charConst \'[a-zA-Z]\'
strConst \"[a-z A-Z]*\"

/*Comments*/
single \/\/(.)*
multi \/\*(.*\n?)*\*\/

/*Operators*/
relOp <|<=|>|>=|==|!=
arithOp "+"|"-"|"*"|"/"|"%"
logicOp &&|\|\||!

/*Separators*/
sep [!@#$^&(){};:,]

```

```
/* Pattern Action pairs*/
%%
{inc} {printf("PREDIR ");}
{relOp} {printf("RELOP ");}
{arithOp} {printf("ARITHOP ");}
{logicOp} {printf("LOGOP ");}
{numConst} {printf("NUMCONST "); set_const(val);}
{charConst} {printf("CHARCONST ");set_const(val);}
{strConst} {printf("STRCONST "); set_const(val);}
{single} {printf("SC ");}
{multi} {printf("MC ");}
{kw} {printf("KW "); set_flag(&flag);}
{funcCall} {printf("FC ");}
{id} {printf("ID "); construct_table(symbol_table, &
   symbol_count);}
{sep} {printf("SP ");}
"=" {printf("ASSIGN ");}
"\n" {printf("\n");}
%%

int yywrap(void){}

void printTable(char *symbol_table[100], int symbol_count){
    for(int i = 0; i<symbol_count;i++){
        char *token = strtok(symbol_table[i], " ");
        while(token){
            printf("%s ", token);
            token = strtok(NULL, " ");
        }
        printf("\n");
    }
}
int main(){
    char *name = (char*)calloc(100, sizeof(char));
    printf("Enter filename: ");scanf(" %[^\n]", name);

    yyin = fopen(name, "r+");
    yylex();

    printTable(symbol_table, symbol_count);
    return 0;
}
```

Input file:

```
#include<stdio.h>
/*Multiline
comment*/
main()
{
  float c = 20;
  int a=10,b=20;

  if (a != b)
      printf(  a is greater );
  else
      printf(  b is greater );
}
add()
{
  int a = 10;
}
//Single line comment
```

Output:



```
Enter filename: file.c
PREDIR
MC
FC
SP
  KW  ID  ASSIGN  NUMCONST SP
  KW  ID ASSIGN NUMCONST SP ID ASSIGN NUMCONST SP

  KW  SP ID  RELOP  ID SP
      FC SP
  KW
      FC SP
SP
FC
SP
  KW  ID  ASSIGN  NUMCONST SP
SP
SC
 Name  Type  Size  Addr Value
    c float     4  1000 20
    a   int     2  1004 10
    b   int     2  1006 20
```

Learning Outcomes:

- Understood the basic working of Lex tool for tokenising.
- Learnt how to construct symbol table from code using lex tool.
