Exercise 2: Lexical Analyser using Lex tool

Praveen Kumar R

January 13, 2020

Assignment	2
Reg No	312217104114
Name	Praveen Kumar R

1 Lex Program

```
%{
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
typedef struct table {
    char identifier[32];
    char type[5];
    int start;
    int size;
    double value;
} Table;
Table t[100];
int t_index = 0,base = 1000,flag = 0,fg[20],cn = 0;
char symbols[20][20],values[20][20];
%}
keyword ("auto"|"break"|"case"|"char"|"const"|"continue"|"default"|
"do"|"double"|"else"|"enum"|"extern"|"float"|"for"|"goto"|"if"|
"int"|"long"|"register"|"return"|"short"|"signed"|"sizeof"|
"static"|"struct"|"switch"|"typedef"|"union"|"unsigned"|"void"|
"volatile"|"while")
function [a-zA-Z_][a-zA-Z0-9_]*[(].*[)]
identifier [a-zA-Z_][a-zA-Z0-9_]*
int_constant [0-9]+
float_constant [0-9]+.[0-9]+
%%
^#.* { printf("%s - preprocessor directive\n", yytext); }
{keyword} {
    int i = 0;
    char s[10]; strcpy(s, yytext);
    while(s[i++] != '\0') if(s[i] == ' ' ||
 s[i] == '\t' || s[i] == '\n') s[i] = ' ';
    printf("%s - keyword\n", s);
    if(strcmp(s,"int")==0) flag = 2;
    else if(strcmp(s,"float")==0) flag  = 4;
}
{function} { printf("%s - function call\n", yytext); }
{identifier} { printf("%s - identifier\n", yytext);
               strcpy(symbols[cn],yytext); }
{int_constant} { printf("%s - integer constant\n", yytext);
               strcpy(values[cn],yytext);
               fg[cn] = flag;
               cn++;
               }
{float_constant} { printf("%s - float/double constant\n", yytext);
               strcpy(values[cn],yytext);
               fg[cn] = flag;
               cn++; }
("<"|"<="|">"|">="|"=="|"!=") { printf("%s - relational operator\n", yytext); }
= { printf("%s - assignment operator\n", yytext); }

[{}(),;] { printf("%s - special character\n", yytext); }
. { }
\n { }
%%
int main(int argc, char* argv[])
{
    yyin = fopen(argv[1], "r");
    yylex();
    printf("SYMBOL TABLE\nTYPE\tSYMBOL\tSIZE\tADDRESS\tVALUE\n");
    for (int i = 0; i<cn ;i++){
        printf("%s\t %s\t %d\t %d\t %s\n",fg[i]==2?"int":"float",
symbols[i],fg[i],base,values[i]);
        base = base+ fg[i];
    }
    return 0;
}
```

2 Output

```
#include<stdio.h> - preprocessor directive
main() - function call
{ - special character
int - keyword
a - identifier
= - assignment operator
10 - integer constant
, - special character
b - identifier
= - assignment operator
20 - integer constant
; - special character
float - keyword
c - identifier
= - assignment operator
10.4 - float/double constant
, - special character
d - identifier
= - assignment operator
20.5 - float/double constant
; - special character
if - keyword
( - special character
a - identifier
> - relational operator
b - identifier
) - special character
printf("a is greater") - function call
; - special character
else - keyword
printf("b is greater") - function call
; - special character
} - special character
SYMBOL TABLE
TYPE SYMBOL SIZE ADDRESS VALUE
int  a  2  1000  10
int  b  2  1002  20
float  c  4  1004  10.4
float  d  4  1008  20.5
```