Exercise 2: Lexical Analyser using Lex Tool

Ram Kaushik R

January 27, 2020

Assignment	2
Reg No	312217104125
Name	Ram Kaushik R

1 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, fg[20], cn = 0;
char symbols[20][20], values[20][20], dtype[20][20];
%}
keyword ("auto"|"break"|"case"|"char"|"const"|"continue"|"default"|"do"|"double"
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;
    strcpy(dtype[cn], "int");
    cn++;
}
{float_constant} {
    printf("%s - float/double constant\n", yytext);
    strcpy(values[cn],yytext);
    fg[cn] = flag;
    strcpy(dtype[cn], "float");
    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();

    for(int i = 0; i < cn; i++) {
printf("%s\t%s\t%d\t%d\t%s\n", symbols[i], dtype[i], fg[i], base, values[i]);
        base += fg[i];
    }
    return 0;
}
```

2 Input

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

```
main()
{
int a=10;
float c=10.3;
int b=20;
if(a>b)
printf("a is greater");
else
printf("b is greater");
}
```

3 Output

```
#include<stdio.h> - preprocessor directive
int - keyword
main() - function call
{ - special character
int - keyword
a - identifier
= - assignment operator
10 - integer constant
; - special character
float - keyword
c - identifier
= - assignment operator
10.3 - float/double constant
; - special character
int - keyword
b - identifier
= - assignment operator
20 - integer constant
; - special character
if(a>b) - function call
printf("a is greater") - function call
; - special character
else - keyword
printf("b is greater") - function call
; - special character
} - special character
a int 2 1000 10
c float 4 1002 10.3
b int 2 1006 20
```