12/02/2021

A2: Implementation of Lexical Analyzer for the patterns using Lex (identifier, comments, operators, constants)

185001188
Vanathi G
CSE C

Program :

```
/*lex program to count number of words*/
%{
	#include<stdio.h>
	#include<string.h>

	typedef struct {
		char type[10];
		char varname[32];
		char init_value[32];

		}symbolTable;

	char curr_type[10];

	symbolTable st[10];
	int ptr = -1, exists = 0, i, assign_expected = 0;

%}

digit [0-9]
letter [a-zA-Z]

digits {digit}+
optFrac \.{digits}
optExp E("+"|"-")?{digits}
numberconst {digits}({optFrac})?({optExp})?

charconst \'{letter}\'
stringconst \"({letter}|" "|{digit})*\"
constant {numberconst}|{charconst}|{stringconst}
```

```
id {letter}({letter}|{digit})*(\[{digit}*\])?

start \/\/
single {start}({letter}|{digit}|" ")*

start1 \/\*
end1 \*\/
multi {start1}({letter}|{digit}|"\n"|" ")*{end1}

relop "<"|"<="|"=="|"!="|">"|">="
arithop "+"|"-"|"*"|"/"|"%"
logicalop "&&"|"||"|"!"
assignop "="

sp ","|";"|"{"|"}"

operator {relop}|{arithop}|{logicalop}|{assignop}

keyword
("auto"|"break"|"case"|"char"|"const"|"continue"|"default"|"do"|"double"|"else"|"e
num"|"extern"|"float"|"for"|"goto"|"if"|"int"|"long"|"register"|"return"|"short"|"sig
ned"|"sizeof"|"static"|"struct"|"switch"|"typedef"|"union"|"unsigned"|"void"|"volatil
e"|"while")

function ("printf"|"main")


/* Rules Section*/
%%
{single} {printf("SINGLE-LINE COMMENT ");}
{multi} {printf("MULTI-COMMENT ");}

{constant} {
	printf("CONST ");
	if(assign_expected == 1)
	{
		strcpy(st[ptr].init_value, yytext);
		assign_expected = 0;
	}
```

```
}

{keyword} {
	printf("KW ");
	if(strcmp(yytext, "int") == 0 || strcmp(yytext, "float") == 0 || strcmp(yytext,
	"double") == 0 || strcmp(yytext, "char") == 0)
		strcpy(curr_type, yytext);
}
{function} {printf("FC ");}
{id} {
	printf("ID ");
	exists = 0;
	for(i=0; i<=ptr; i++)
	{
		if(strcmp(st[i].varname, yytext) == 0)
			exists = 1;
	}
	if(exists == 0)
	{
		ptr++;
		strcpy(st[ptr].type, curr_type);
		strcpy(st[ptr].varname, yytext);
		strcpy(st[ptr].init_value, "");
	}

}

{operator} {
	printf("OP ");
	if(yytext[0] == '=')
		assign_expected = 1;

}
{sp} {printf("SP ");}

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

[" "|"("|")"] {};
%%
```

```
int yywrap(void){return 1;}

int main()
{
   // The function that starts the analysis
   yylex();

   printf("\n--------------------\nID\tType\tValue\n");
   for(int i=0; i<=ptr; i++)
   {
         printf("%s\t%s\t%s\n", st[i].varname, st[i].type, st[i].init_value);

   }

   return 0;
}
```

I/O Snapshot -