Compiler Design Integrated Lab
A Anirudh
195001015
CSE-A

Programming Assignment-1 - Implementation of lexical analyser and symbol table

In the first programming assignment, you will get your compiler off to a great start by implementing Lexical analyzer or Scanner using C. Your scanner will run through the source program recognizing C tokens in the order in which they are read, until end of file is reached. When an identifier is encountered, it should be stored in symbol table with its attributes. Symbol table consist of the attributes identifier name, type, no of bytes, location and value. Your scanner should identify the tokens categorized below and print it.

Code:

prgm.c

```
#include<stdio.h>
#include<string.h>
int main()
{
   int a=10,b=20;
   char c;
   float d = 1.111;
   if(a>b)
        printf("a is greater");
   else
        printf("b is greater");
}
```

ex1.c

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<ctype.h>
struct table
{
	char id[10];
	char type[10];
	int size;
	int add;
	char val[10];
};  //For Symbol Table
int iskey(char str[20])
{
	char key[32]
[20]={"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"};
	for(int i=0;i<32;i++)
	{
		if(strcmp(str,key[i])==0)
		{
			return 1;
		}
	}
	return 0;
}
char words[15][31]={"preprocessor directive","function call","special
character","keyword","identifier","assignment operator","integer
constant","arithmetic operator","logical operator","relational operator","bitwise
operator","unary operator","arithmetic assignment operator"};
char sp[10][2]={";",",","(",")","{","}"};
int main()
{
	struct table t[10];
	int x=0,y=0;
	char str[50],path[50];
	char ch;
	FILE *fp;
	printf("\n Enter file name with path :");
	scanf("%s",path);
	fp=fopen(path,"r"); //To open a file
	if(fp==NULL)
	{
		printf("\n There is no file");
		exit;
	}
	else
   {
		while (fscanf(fp,"%[^\n] ",str) != EOF)
		{
			char s[20]="";
			char s1[10]="";
			char s2[10]="";
```

```
                        for(int j=0;str[j]!='\0',j<strlen(str);j++)
                        {
                                if(str[j]=='#')
                                {
                                        printf("\n%s - %s",str,words[0]);
                                        break;
                                }
                                else if(str[j]=='{' || str[j]=='}')
                                {
                                        printf("\n%s - %s",str,words[2]);
                                        continue;
                                }
                                else if(j==strlen(str)-1 && (str[j]==';' || str[j]==']' ||
str[j]==')' || str[j]=='}' ))
                                {
                                        printf("\n%c - %s",str[j],words[2]);
                                        continue;
                                }
                                else if(isdigit(str[j]) && !isdigit(str[j+1]))
                                {
                                        strncat(s1,&str[j],1);
                                        strcpy(t[x-1].val,s1);
                                        printf("\n%d - %s",atoi(s1),words[6]);
                                        strcpy(s1,"");
                                        continue;
                                }
                                else if(isdigit(str[j]) && isdigit(str[j+1]))
                                {
                                        strncat(s1,&str[j],1);
                                        continue;
                                }
                                else if(str[j]=='=')
                                {
                                        printf("\n%c - %s",str[j],words[5]);
                                        continue;
                                }
                                else if(isalnum(str[j]) && !isalnum(str[j+1]))
                                {
                                        strncat(s,&str[j],1);
                                        if(iskey(s)==1)
                                        {
                                                printf("\n%s - %s",s,words[3]);
                                                strcpy(t[x].type,s);
                                        }
                                        else if(strcmp(s,"main")==0)
                                        {
                                                printf("\n%s() - %s",s,words[1]);
                                                break;
                                        }
                                        else if((strcmp(s,"printf")==0) ||
(strcmp(s,"scanf")==0) || (strcmp(s,"getch")==0) || (strcmp(s,"clrscr")==0))
                                        {
                                                char str1[80]="";
                                                for(int k=0;str[k]!='\0';k++)
                                                {
                                                        if(str[k]!=';')
```

```
                                        {
                                                strncat(str1,&str[k],1);
                                        }
                                }
                                printf("\n%s - %s",str1,words[1]);
                                printf("\n; - %s",words[2]);
                                break;
                        }
                        else
                        {
                                printf("\n%s - %s",s,words[4]);
                                strcpy(t[x++].id,s);
                                strcpy(t[x].type,t[x-1].type);
                        }
                        strcpy(s,"");
                        continue;
                }
                else if(isalnum(str[j]) && isalnum(str[j+1]))
                {
                        strncat(s,&str[j],1);
                        continue;
                }
                else if(!isalnum(str[j]) && !isalnum(str[j+1]))
                {
                        strncat(s2,&str[j],1);
                        strncat(s2,&str[j],1);
                        if((strcmp(s2,"+=")==0) || (strcmp(s2,"-=")==0) ||
(strcmp(s2,"*=")==0) || (strcmp(s2,"/=")==0)|| (strcmp(s2,"%=")==0))
                        {
                                printf("\n%s - %s",s2,words[12]);
                        }
                        else if((strcmp(s2,"&&")==0) || (strcmp(s2,"||")==0)
|| (strcmp(s2,"*=")==0))
                        {
                                printf("\n%s - %s",s2,words[8]);
                        }
                        else if((strcmp(s2,"<=")==0) || (strcmp(s2,">=")==0)
|| (strcmp(s2,"==")==0) || (strcmp(s2,"!=")==0))
                        {
                                printf("\n%s - %s",s2,words[9]);
                        }
                        else if((strcmp(s2,">>")==0) ||
(strcmp(s2,"<<")==0))
                        {
                                printf("\n%s - %s",s2,words[10]);
                        }
                        else if((strcmp(s2,"++")==0) || (strcmp(s2,"--")==0))
                        {
                                printf("\n%s - %s",s2,words[11]);
                        }
                        continue;
                }
                else if(!isalnum(str[j])) //Checks
                {
                        if( str[j]=='+' || str[j]=='-' || str[j]=='*' || str[j]=='/')
                        {
```

```
                                        printf("\n%c - %s",str[j],words[7]);
                                }
                                else if( str[j]=='!')
                                {
                                        printf("\n%c - %s",str[j],words[8]);
                                }
                                else if( str[j]=='<' || str[j]=='>')
                                {
                                        printf("\n%c - %s",str[j],words[9]);
                                }
                                else if( str[j]=='^' || str[j]=='&' || str[j]=='|')
                                {
                                        printf("\n%c - %s",str[j],words[10]);
                                }
                                else if( str[j]==';' || str[j]==',' || str[j]=='.' ||
str[j]=='[' || str[j]==']' || str[j]=='(' || str[j]==')' || str[j]=='{' || str[j]=='}')
                                {
                                        printf("\n%c - %s",str[j],words[2]);
                                }
                                continue;
                        }
                }
        }
        int z1=1000;
        printf("\n\nContents of symbol table:\n\n");
        printf("%-20s %-20s %-20s %-20s %-20s \n","identifier name","type","no. of
bytes","address","value");
        for(int i=0;i<x;i++)
        {
                if(strcmp(t[i].type,"int")==0)
                {
                        strcpy(t[i].type,"int");
                        t[i].size=sizeof(int);
                        t[i].add=z1;
                        z1=z1+sizeof(int);
                        if(!isalnum(t[i].val[0]) && !isalnum(t[i].val[1]))
                        {
                                strcpy(t[i].val,"\0");
                        }
                        printf("|%-20s|%-20s|%-20d|%-20d|%-
20s|\n",t[i].id,t[i].type,t[i].size,t[i].add,t[i].val);

                }
                else if(strcmp(t[i].type,"float")==0)
                {
                        strcpy(t[i].type,"float");
                        t[i].size=sizeof(float);
                        t[i].add=z1;
                        z1=z1+sizeof(float);
                        if(!isalnum(t[i].val[0]))
                        {
                                t[i].val[0]='\0';
                        }
                        printf("|%-20s|%-20s|%-20d|%-20d|%-
20s|\n",t[i].id,t[i].type,t[i].size,t[i].add,t[i].val);
```

```
                }
                else if(strcmp(t[i].type,"char")==0)
                {
                        strcpy(t[i].type,"char");
                        t[i].size=sizeof(char);
                        t[i].add=z1;
                        z1=z1+sizeof(char);
                        if(!isalnum(t[i].val[0]))
                        {
                                t[i].val[0]='\0';
                        }
                        printf("|%-20s|%-20s|%-20d|%-20d|%-
20s|\n",t[i].id,t[i].type,t[i].size,t[i].add,t[i].val);

                }
        }
        printf("\n\n");
        return 0;
}
```

Output:

```
cduser1@sel12-HP-Compaq-Pro-6305-SFF:~/Desktop$ gcc -o ex ex1.c
cduser1@sel12-HP-Compaq-Pro-6305-SFF:~/Desktop$ ./ex

 Enter file name with path :prgm.c

#include<stdio.h> - preprocessor directive
#include<string.h> - preprocessor directive
int - keyword
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
char - keyword
c - identifier
; - special character
float - keyword
d - identifier
= - assignment operator
1 - integer constant
. - special character
111 - integer 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
```

```
Contents of symbol table:

identifier name      type                no. of bytes        address             value
|a                   |int                |4                  |1000               |10                 |
|b                   |int                |4                  |1004               |20                 |
|c                   |char               |1                  |1008               |                   |
|d                   |float              |4                  |1009               |111                |
```