Expt. No. 1 Page No. 1

Expt. Name. Implement Token seperation for a given expression Date :

Aim:

To write a lex program to implement Token seperation for a given expression using LEX.

Algorithm:

step 1: Include the necessary header files and declare the necessary variables

step 2: Define the key words and the identifiers with the constant and operator

step 3: Get the statement for analysis from the user

step 4:: check ~~the~~ each and every element in the statement with the define keyword and if it matches print it as the keyword.

step 5: check each and every element in the statement with the defined identifier and if it matches print the element as an identifier.

step 6: check each and every element in the statement with the defined constant then find the equivalance and ~~point~~ print it as constant

step 7: check each and every element in the statement with the defined operator and if it matches print the element as an identifier

step 8: Display every element on the screen as seperated

step 9: Exit the program

Expt. No. _____ Page No. 2

Expt. Name. _____ Date : _____

Program

File.l

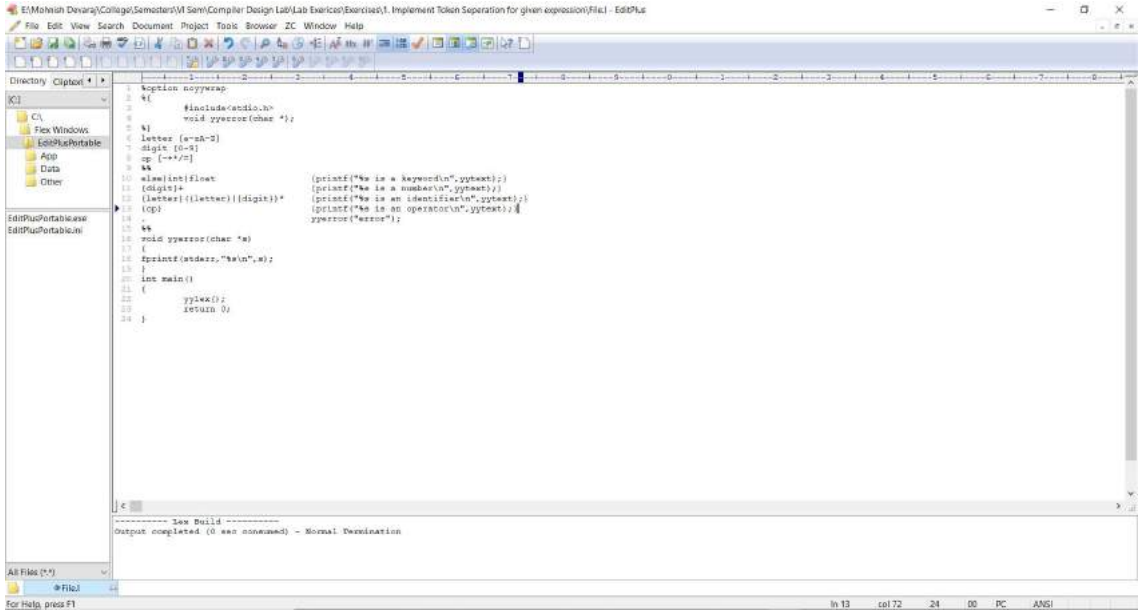
```
letter %option noyywrap
%{
    #include<stdio.h>
    void yyerror(char *);
%}
letter [A-Za-z]
digit [0-9]
operator [-+*]
%%
void|
main|
if|
do|
printf|
int|
float|
char|
for                {printf("%s is a keyword\n",yytext);}
"%s"|
"%c"|
"%d"|
"%f"               {printf("%s is a Format Specifier\n",yytext);}
({letter}|_)({letter}|{digit})*   {printf("%s is a identifier\n",yytext);}
{operator}({operator})*           {printf("%s is an operator\n",yytext);}
{digit}+           {printf("%s is a number\n",yytext);}
"("                {printf("%s is a open parenthesis\n",yytext);}
```

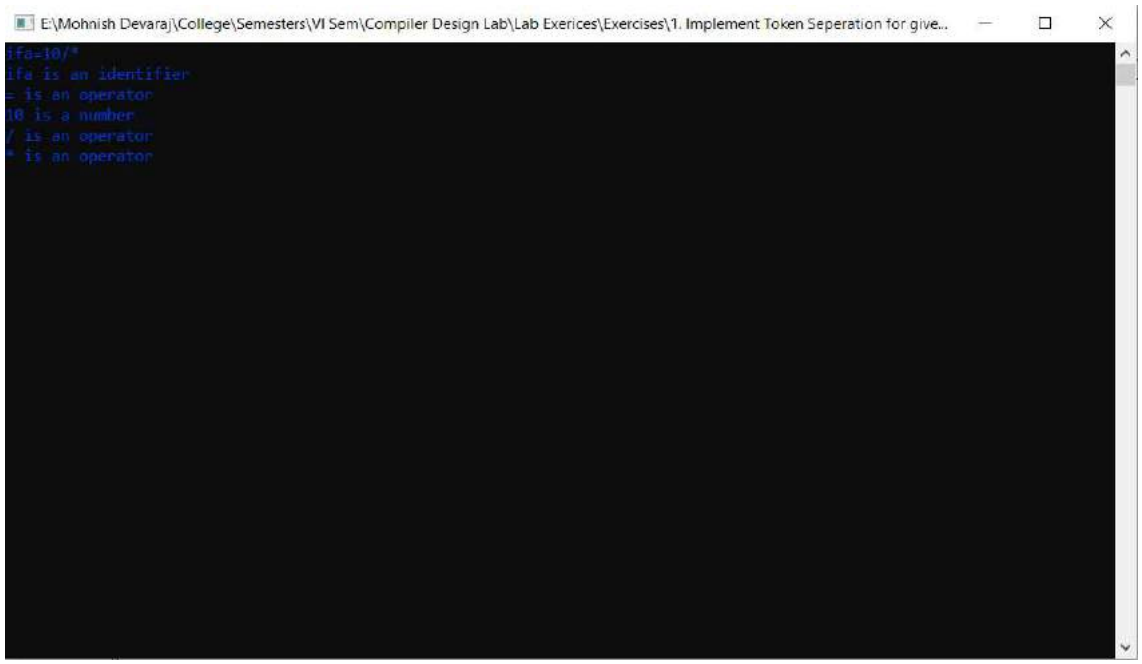
```
)          {printf("%s is an close parenthesis\n",yytext);}
;          {printf("%s is a semicolon\n",yytext);}
\{
\}
\=
\/
\,
\'
\"         {printf("%s is a double quote\n",yytext);}
.          {printf("\n Syntax Error!!\n");}
%%

void yyerror(char *s)
{
fprintf(stderr,"%s\n",s);
}
int main()
{
  yylex();
  return 0;
}
```

Result:

Implement Token seperation for given expression using LEX TOOL has been executed successfully.


E:\Mohnish Devaraj\College\Semesters\VI Sem\Compiler Design Lab\Lab Exercices\Exercises\1. Implement Token Seperation for given expression\File1 - EditPlus
File Edit View Search Document Project Tools Browser ZC Window Help
Directory Clipbook
C:\
Flex Windows
EditPlusPortable
App
Data
Other
EditPlusPortable.exe
EditPlusPortable.ini
All Files (*.*)
File1
%option noyywrap
%{
#include<stdio.h>
void yyerror(char *);
%}
letter [a-zA-Z]
digit [0-9]
op [-+*/=]
%%
else|int|float (printf("%s is a keyword\n",yytext);)
{digit}+ (printf("%s is a number\n",yytext);)
{letter}({letter}|{digit})* (printf("%s is an identifier\n",yytext);)
{op} (printf("%s is an operator\n",yytext);)
. yyerror("error");
%%
void yyerror(char *s)
{
fprintf(stderr,"%s\n",s);
}
int main()
{
yylex();
return 0;
}
---------- lex Build ----------
Output completed (0 sec consumed) - Normal Termination
For Help, press F1
ln 13 col 72 24 00 PC ANSI


E:\Mohnish Devaraj\College\Semesters\VI Sem\Compiler Design Lab\Lab Exercices\Exercises\1. Implement Token Seperation for give...
ifa=10/*
ifa is an identifier
= is an operator
10 is a number
/ is an operator
* is an operator