Exp 2 - Lexical Analyzer
RA1911003010143
Abhishek Kumar

Exp 2 - Lexical Analyzer

Aim: To implement a lexical analyzer.

Algorithm:

1. Tokenization i.e. Dividing the program into valid tokens.
2. Remove white space characters.
3. Remove comments.
4. It also provides help in generating error messages by providing row numbers and column
numbers.

Program:

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
void keyw(char *p);
int i = 0, id = 0, kw = 0, num = 0, op = 0, sep = 0;
char keys[32][10] = {"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"};
main()

{
    char ch, str[25], seps[15] = " \t\n,;(){}[]#\"<>", oper[] = "!%^&*-
+=~|.<>/?";
    int j;
    char fname[50]= "C:\\Users\\abhis\\Downloads\\file.txt";
    FILE *f1;

    f1 = fopen(fname, "r");
    if (f1 == NULL)
```

Exp 2 - Lexical Analyzer
RA1911003010143
Abhishek Kumar

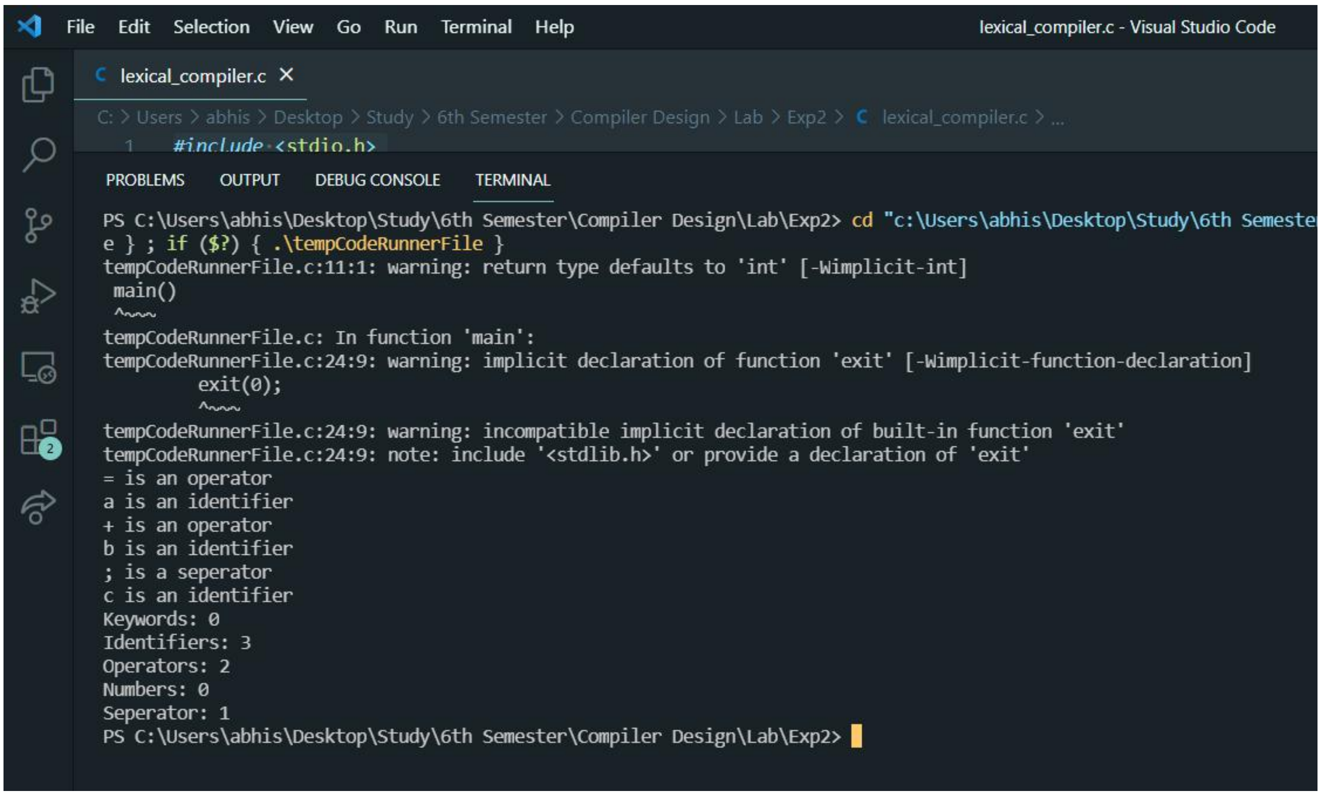
```c
    {
        printf("file not found");
        exit(0);
    }
    while ((ch = fgetc(f1)) != EOF)
    {
        for (j = 0; j <= 14; j++)
        {
            if (ch == oper[j])
            {
                printf("%c is an operator\n", ch);
                op++;
                str[i] = '\0';
                keyw(str);
            }
        }
        for (j = 0; j <= 14; j++)
        {
            if (i == -1)
                break;
            if (ch == seps[j])
            {
                if (ch == '#')
                {
                    while (ch != '>')
                    {
                        printf("%c", ch);
                        ch = fgetc(f1);
                    }
                    printf("%c is a header file\n", ch);
                    i = -1;
                    break;
                }
                if (ch ==';')
                {
                    sep++;
                    printf("%c is a seperator\n", ch);
                }
                if (ch == '"')
                {
                    do
                    {
                        ch = fgetc(f1);
                        printf("%c", ch);
                    } while (ch != '"');
                    printf("\b is an argument\n");
                    i = -1;
                    break;
                }
                str[i] = '\0';
                keyw(str);
            }
        }
        if (i != -1)
        {
            str[i] = ch;
            i++;
        }
        else
            i = 0;
    }
    printf("Keywords: %d\nIdentifiers: %d\nOperators: %d\nNumbers: %d\n
Seperator: %d\n", kw, id, op, num, sep);
    //getch();
}
void keyw(char *p)
{
    int k, flag = 0;
    for (k = 0; k <= 31; k++)
    {
        if (strcmp(keys[k], p) == 0)
        {
            printf("%s is a keyword\n", p);
            kw++;
            flag = 1;
            break;
        }
    }
    if (flag == 0)
    {
        if (isdigit(p[0]))
        {
            printf("%s is a number\n", p);
            num++;
        }
        else
        {
            if (p[0] != '\0')
            {
                printf("%s is an identifier\n", p);
                id++;
            }
        }
    }
    i = -1;
}
```

file.txt:

a=b+c;

Exp 2 - Lexical Analyzer
RA1911003010143
Abhishek Kumar

Output:



File Edit Selection View Go Run Terminal Help
lexical_compiler.c - Visual Studio Code
lexical_compiler.c
C: > Users > abhis > Desktop > Study > 6th Semester > Compiler Design > Lab > Exp2 > lexical_compiler.c > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

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

```
PS C:\Users\abhis\Desktop\Study\6th Semester\Compiler Design\Lab\Exp2> cd "c:\Users\abhis\Desktop\Study\6th Semeste
e } ; if ($?) { .\tempCodeRunnerFile }
tempCodeRunnerFile.c:11:1: warning: return type defaults to 'int' [-Wimplicit-int]
 main()
 ^~~~
tempCodeRunnerFile.c: In function 'main':
tempCodeRunnerFile.c:24:9: warning: implicit declaration of function 'exit' [-Wimplicit-function-declaration]
         exit(0);
         ^~~~
tempCodeRunnerFile.c:24:9: warning: incompatible implicit declaration of built-in function 'exit'
tempCodeRunnerFile.c:24:9: note: include '<stdlib.h>' or provide a declaration of 'exit'
= is an operator
a is an identifier
+ is an operator
b is an identifier
; is a seperator
c is an identifier
Keywords: 0
Identifiers: 3
Operators: 2
Numbers: 0
Seperator: 1
PS C:\Users\abhis\Desktop\Study\6th Semester\Compiler Design\Lab\Exp2>
```

Result: Program to implement a lexical analyzer was written and executed successfully.