

RA2011031010046

EXPERIMENT : 1

DATE : 18/01/23

LEXICAL ANALYSER

AIM : Write a program to implement Lexical Analyser

PROCEDURE:

1. We create a text file named "input_file.txt" where the source program is present.
2. We create a user defined function called "isKeyword" to check for keywords and return the value for the flag variable.
3. In the main function we create a file pointer to read from the input file.
4. We check whether the character is an operator, keyword or an identifier character by character.
5. We create a count variable and increment it to count the number of tokens in the input file.
6. We then display the token count.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
int isKeyword(char temp[])
{
    char keywords[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"};
    int i, flag = 0;
    for (i = 0; i < 32; ++i)
    {
        if (strcmp(keywords[i], temp) == 0)
        {
            flag = 1;
            break;
        }
    }
    return flag;
}
int main()
{
    char ch, temp[15], opr[] = "+-*/%=";
    FILE *fp;
```

```

int i, j = 0, count=0;;
fp = fopen("input_file.txt", "r");
if (fp == NULL)
{
    printf("Sorry, cannot open the file!! Try again!\n");
    exit(0);
}
while ((ch = fgetc(fp)) != EOF)
{
    for (i = 0; i < 6; ++i)
    {
        if (ch == opr[i])
        { printf("%c is an OPERATOR\n", ch);
          count++;
        }
    }
    if (isalnum(ch))
    {
        temp[j++] = ch;
    }
    else if ((ch == ' ' || ch == '\n') && (j != 0))
    {
        temp[j] = '\0';
        j = 0;
        if (isKeyword(temp) == 1)
        { printf("%s is a KEYWORD\n", temp);
          count++;
        }
        else
        { printf("%s is an IDENTIFIER\n", temp);
          count++;
        }
    }
}
printf("\nThe number of tokens present in the given file are: %d", count);
fclose(fp);
return 0;
}

```

INPUT :

“input_file.txt”

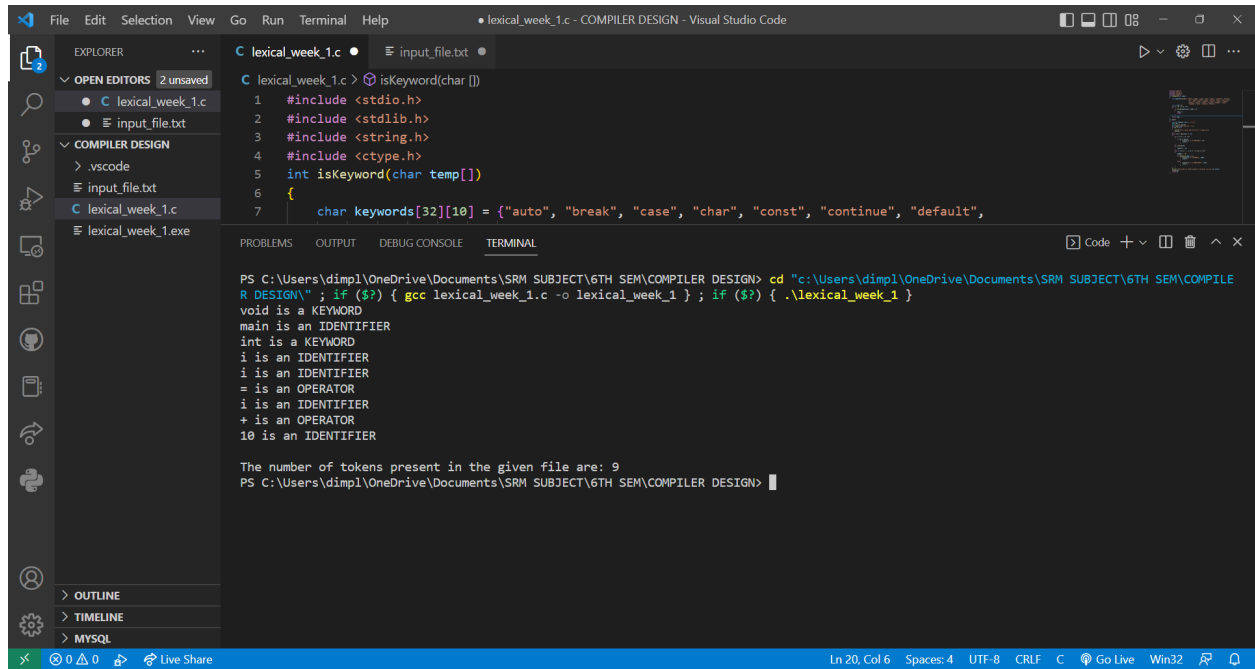
```

void main()
{
    int i;

```

```
i = i + 10; }
```

OUTPUT:



The screenshot shows the Visual Studio Code interface with the following components:

- EXPLORER:** Shows the file structure with 'lexical_week_1.c' and 'input_file.txt' under 'COMPILER DESIGN'.
- EDITOR:** Displays the C code for 'lexical_week_1.c'. The code includes standard headers, defines a keyword list, and implements a function to check if a character is a keyword.
- TERMINAL:** Shows the command prompt output. The user runs 'gcc lexical_week_1.c -o lexical_week_1' and then executes the program. The output lists the tokens found in 'input_file.txt': 'void', 'is', 'a', 'KEYWORD', 'main', 'is', 'an', 'IDENTIFIER', 'int', 'is', 'a', 'KEYWORD', 'i', 'is', 'an', 'IDENTIFIER', 'i', 'is', 'an', 'IDENTIFIER', '=', 'is', 'an', 'OPERATOR', 'i', 'is', 'an', 'IDENTIFIER', '+', 'is', 'an', 'OPERATOR', and '10', 'is', 'an', 'IDENTIFIER'. It concludes with 'The number of tokens present in the given file are: 9'.

RESULT:

Hence, the Lexical Analyser program was successfully implemented.