EX.NO.1.a

Date:

**Implementation of lexical analyzer using C**

**AIM**

To Implement lexical analyzer using C program.

**ALGORITHM**

**Step 1:**Start.

**Step 2:**Include the necessary header files:

* stdio.h for input/output functions.
* string.h for string manipulation functions.
* ctype.h for character handling functions.

**Step 3:**Define the function int isKeyword(char buffer[]) to check if a given string is a keyword.

* **Step 3.1:** Initialize an array of 32 C language keywords.
* **Step 3.2:** Iterate through the keyword array.
* **Step 3.3:** Compare the given string (buffer) with each keyword.
* **Step 3.4:** If a match is found, return 1 (indicating it is a keyword).
* **Step 3.5:** If no match is found, return 0.

**Step 4:**Define the function int isOperator(char ch) to check if a given character is an operator.

* **Step 4.1:** Initialize an array of common C language operators.
* **Step 4.2:** Iterate through the operator array.
* **Step 4.3:** Compare the given character (ch) with each operator.
* **Step 4.4:** If a match is found, return 1 (indicating it is an operator).
* **Step 4.5:** If no match is found, return 0.

**Step 5:**Define the function void lexicalAnalyzer(FILE \*fp) to perform lexical analysis on the input file.

* **Step 5.1:** Initialize a character buffer and an index variable.
* **Step 5.2:** Open a while loop to read the file character by character until the end of the file is reached.
  + **Step 5.2.1:** If the character is alphanumeric, add it to the buffer.
  + **Step 5.2.2:** If the character is a space, newline, or operator, and the buffer is not empty:
    - Null-terminate the buffer.
    - Check if the buffer contains a keyword, number, or identifier, and print the result.
    - Reset the buffer index.
  + **Step 5.2.3:** If the character is an operator, print it.
  + **Step 5.2.4:** If the character is % or ", handle it as a format specifier or argument in printf/scanf and print the result.
* **Step 5.3:** End the while loop.

**Step 6:**Define the main() function as the entry point of the program.

* **Step 6.1:** Declare a file pointer and a character array for the filename.
* **Step 6.2:** Read the filename from the user.
* **Step 6.3:** Open the file in read mode.
* **Step 6.4:** If the file cannot be opened, print an error message and terminate the program.
* **Step 6.5:** Call the lexicalAnalyzer(fp) function to analyze the file.
* **Step 6.6:** Close the file.
* **Step 6.7:** Return 0 to indicate successful execution.

**Step 7:**Stop.

**PROGRAM**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

int isKeyword(char buffer[]) {

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;

for (i = 0; i < 32; ++i) {

if (strcmp(keywords[i], buffer) == 0) {

return 1;

}

}

return 0;

}

int isOperator(char ch) {

char operators[] = "+-\*/%=";

for (int i = 0; i < strlen(operators); ++i) {

if (ch == operators[i]) {

return 1;

}

}

return 0;

}

void lexicalAnalyzer(FILE \*fp) {

char ch, buffer[15];

int i = 0;

while ((ch = fgetc(fp)) != EOF) {

// Identifying keywords and identifiers

if (isalnum(ch)) {

buffer[i++] = ch;

} else if ((ch == ' ' || ch == '\n' || isOperator(ch)) && i != 0) {

buffer[i] = '\0';

i = 0;

if (isKeyword(buffer)) {

printf("%s is a keyword\n", buffer);

} else if (isdigit(buffer[0])) {

printf("%s is a number\n", buffer);

} else {

printf("%s is an identifier\n", buffer);

}

}

// Identifying operators

if (isOperator(ch)) {

printf("%c is an operator\n", ch);

}

// Handling arguments in printf/scanf

if (ch == '%' || ch == '"') {

if (ch == '%') {

printf("%c is an argument\n", ch);

}

while ((ch = fgetc(fp)) != '"' && ch != EOF) {

printf("%c", ch);

}

printf(" is an argument\n");

}

}

}

int main() {

FILE \*fp;

char filename[100];

scanf("%s", filename);

fp = fopen(filename, "r");

if (fp == NULL) {

printf("Error: Cannot open file %s\n", filename);

return 1;

}

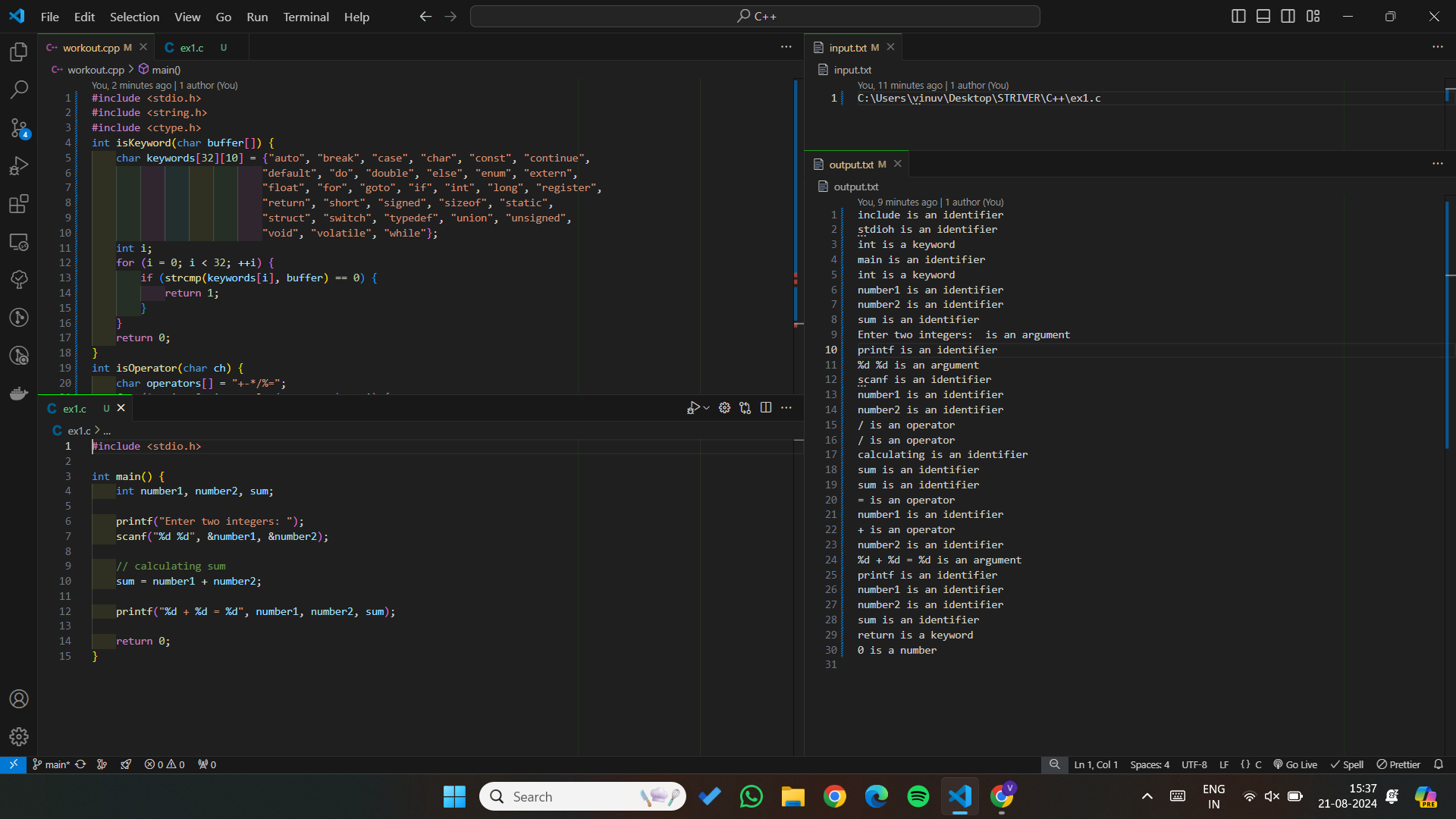
lexicalAnalyzer(fp);

fclose(fp);

return 0;

}

**OUTPUT**

****

**RESULT**

Thus to Implement lexical analyzer using C program has been executed and verified successfully.