4.Design a lexical Analyzer to validate operators to recognize the operators +,-,\*,/ using regular arithmetic operators using C?

**Program:**

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#define MAX\_LEXEME\_LEN 100

void analyze(char \*lexeme) {

if (isalpha(lexeme[0])) {

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

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

printf("%s is a constant\n", lexeme);

} else if (strcmp(lexeme, "+") == 0 || strcmp(lexeme, "-") == 0 ||

strcmp(lexeme, "\*") == 0 || strcmp(lexeme, "/") == 0) {

printf("%s is an operator\n", lexeme);

} else {

printf("%s is an unknown token\n", lexeme);

}

}

int is\_comment(char \*str, int \*index) {

if (str[\*index] == '/' && str[\*index + 1] == '\*') {

\*index += 2;

while (str[\*index] != '\0' && !(str[\*index] == '\*' && str[\*index + 1] == '/')) {

(\*index)++;

}

if (str[\*index] == '\*') {

\*index += 2;

}

return 1;

}

return 0;

}

int main() {

char input[1000], lexeme[MAX\_LEXEME\_LEN];

int i = 0, j = 0;

printf("Enter the input string: ");

fgets(input, 1000, stdin);

while (input[i] != '\0') {

if (isspace(input[i])) {

if (j != 0) {

lexeme[j] = '\0';

analyze(lexeme);

j = 0;

}

} else if (is\_comment(input, &i)) {

continue;

} else {

lexeme[j++] = input[i];

}

i++;

}

if (j != 0) {

lexeme[j] = '\0';

analyze(lexeme);

}

return 0;

}

