**PROGRAM 1:**

1. Develop a lexical Analyzer to identify identifiers, constants, operators using C program

CODE: #include <stdio.h>

#include <ctype.h>

#include <string.h>

#define MAX\_TOKEN\_LENGTH 100

void identifyTokens(char \*code) {

char token[MAX\_TOKEN\_LENGTH];

int i = 0, j = 0;

printf("Tokens Identified:\n");

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

// Ignore whitespace

if (isspace(code[i])) {

i++;

continue;

}

// Handle operators

if (strchr("+-\*/=%><", code[i])) {

printf("Operator: %c\n", code[i]);

i++;

continue;

}

// Handle constants (numbers)

if (isdigit(code[i])) {

while (isdigit(code[i]) || code[i] == '.') {

token[j++] = code[i++];

}

token[j] = '\0';

printf("Constant: %s\n", token);

j = 0;

continue;

}

// Handle identifiers (variables or keywords)

if (isalpha(code[i]) || code[i] == '\_') {

while (isalnum(code[i]) || code[i] == '\_') {

token[j++] = code[i++];

}

token[j] = '\0';

printf("Identifier: %s\n", token);

j = 0;

continue;

}

// Handle special characters or unknown symbols

printf("Unknown Symbol: %c\n", code[i]);

i++;

}

}

int main() {

// Hardcoded input code

char code[] = "int main() { int a = 10; float b = 20.5; a = a + b; }";

printf("Input Code:\n%s\n\n", code);

identifyTokens(code);

return 0;

}

OUTPUT:

Input Code:

int main() { int a = 10; float b = 20.5; a = a + b; }

Tokens Identified:

Identifier: int

Identifier: main

Unknown Symbol: (

Unknown Symbol: )

Unknown Symbol: {

Identifier: int

Identifier: a

Operator: =

Constant: 10

Unknown Symbol: ;

Identifier: float

Identifier: b

Operator: =

Constant: 20.5

Unknown Symbol: ;

Identifier: a

Operator: =

Identifier: a

Operator: +

Identifier: b

Unknown Symbol: ;

Unknown Symbol: }