EXP NO: 2 **DATE: 19.02.25**

DEVELOP A C PROGRAM TO ANALYZE A GIVEN C CODE SNIPPET AND RECOGNIZE DIFFERENT TOKENS, INCLUDING KEYWORD, IDENTIFIERS, OPERATOR AND SPECIAL SYMBOLS

AIM:

To develop a C program that analyzes a given C code snippet and recognizes different tokens, including keywords, identifiers, operators, and special symbols.

ALGORITHM:

- **Start**
- Take a C code snippet as input from the user or a file.
- Initialize necessary arrays and variables for keywords, identifiers, operators, and special symbols.
- Tokenize the input string using spaces, newlines, and other delimiters.
- For each token:
  - Check if it is a **keyword** (compare with a predefined list of C keywords).
  - Check if it is an **identifier** (valid variable/function name that doesn't match a keyword).
  - Check if it is an **operator** (e.g., +, -, *, /, ==, &&).
  - Check if it is a **special symbol** (e.g., {, }, (,), ;, ,).
- Print the categorized tokens.
- **End**

PROGRAM:

```
#include <stdio.h>

#include <string.h>

#include <ctype.h>

const char *keywords[] = {

    "int", "float", "char", "double", "if", "else", "for", "while",

    "do", "return", "void", "switch", "case", "break", "continue",

    "default", "struct", "typedef", "enum", "union", "static",

    "extern", "const", "sizeof", "goto", "volatile", "register"

};

const int num_keywords = sizeof(keywords) / sizeof(keywords[0]);
```

```
const char *operators[] = {"+", "-", "*", "/", "=", "==", "!=", "<", ">", "<=", ">=", "&&", "||",
"++", "--"};
const int num_operators = sizeof(operators) / sizeof(operators[0]);
const char special_symbols[] = {';', '(', ')', '{', '}', '[', ']', ',', '#', '&', '|', ':', '"', '\'};
int isKeyword(char *word) {
    for (int i = 0; i < num_keywords; i++) {
        if (strcmp(word, keywords[i]) == 0)
            return 1;
    }
    return 0;
}
 int isOperator(char *word) {
    for (int i = 0; i < num_operators; i++) {
        if (strcmp(word, operators[i]) == 0)
            return 1;
    }
    return 0;
}
int isSpecialSymbol(char ch) {
    for (int i = 0; i < sizeof(special_symbols); i++) {
        if (ch == special_symbols[i])
            return 1;
    }
    return 0;
}
void analyzeTokens(char *code) {
    char *token = strtok(code, " \t\n"); // Tokenizing by spaces, tabs, and newlines
    printf("\nRecognized Tokens:\n");
    while (token != NULL) {
        if (isKeyword(token))
            printf("Keyword: %s\n", token);
        else if (isOperator(token))
            printf("Operator: %s\n", token);
```

```
        else if (isalpha(token[0]) || token[0] == '_')
         printf("Identifier: %s\n", token);
        else if (isSpecialSymbol(token[0]))
            printf("Special Symbol: %s\n", token);
        else
            printf("Unknown Token: %s\n", token);
        token = strtok(NULL, " \t\n");
    }
}
int main() {
    char code[500];
    printf("Enter a C code snippet:\n");
    fgets(code, sizeof(code), stdin);
    analyzeTokens(code);
    return 0;
}
```

OUTPUT:

```
Enter a C code snippet:
int main() {
    int a = 5, b = 10;
    float c = a + b;
    if (c > 10) {
        printf("Result: %f", c);
    }
    return 0;
}


Recognized Tokens:
Keyword: int
Identifier: main()
Special Symbol: {
```

RESULT :

Thus the above program reads a C code snippet, tokenizes it using space, tab, and newline as delimiters, classifies each token as a keyword, identifier, operator, or special symbol based on predefined lists, and prints the recognized tokens along with their types.