

**EXP NO: 02**

**DATE:**

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

**AIM:**

To develop a C program that analyses a given C code snippet and recognizes different tokens, including keywords, identifiers, operators, delimiter 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>

int main() {
    char input[100];
    char *str[] = {"int","float","long","double","printf"};
    int i=0,j=0,iskeyword=0;
    scanf("%[^END]s",input);
```

```

for(i=0;i<4;i++){
    int flag=1;
    for(j=0;str[i][j]!='\0';j++){
        if(input[j]!=str[i][j]){
            flag=0;
            break;
        }
    }
    if(flag) {
        iskeyword = 1;
        printf("%s is a keyword\n", str[i]);
        break;
    }
}

```

```

int start = j;
while(input[start]!='\0'){
    if(isalpha(input[start])){
        printf("%c",input[start]);
        start++;
        while(isalnum(input[start]) || input[start]=='_'){
            printf("%c",input[start]);
            start++;
        }
        printf(" is a identifier\n");
    }else if(isdigit(input[start])){
        printf("%c",input[start]);
        start++;
        while(isdigit(input[start])){
            printf("%c",input[start]);
            start++;
        }
        printf(" is a constant\n");
    }else if(input[start]=='.' || input[start]==';'){
        printf("%c is a delimiter\n",input[start]);
    }
}

```

```

        start++;
    } else if(input[start]=='+' || input[start]=='-' || input[start]=='*' || input[start]=='/' || input[start]=='%' ||
input[start]=='='){
        printf("%c is a operator\n",input[start]);
        start++;
    } else if(input[start]=='(' || input[start]==')' || input[start]=='{' || input[start]=='}' || input[start]=='[' ||
input[start]==']'){
        printf("%c is a Symbol\n",input[start]);
        start++;
    } else {
        start++;
    }
}
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: {

```

<b>Implementation</b>	
<b>Output/Signature</b>	

**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