Ex-2-GENERATION OF LEXICAL TOKENS LEX FLEX TOOL

AIM

To write a lex program to implement lexical analyzer to recognize a few patterns.

ALGORITHM

1. Start the program.
2. Lex program consists of three parts.
   a. Declaration %%
   b. Translation rules %%
   c. Auxilary procedure.
3. The declaration section includes declaration of variables, maintest, constants and regular definitions.
4. Translation rule of lex program are statements of the form
   a. P1 {action}
   b. P2 {action}
   c. ...
   d. ...
   e. Pn {action}
5. Write a program in the vi editor and save it with .l extension.
6. Compile the lex program with lex compiler to produce output file as lex.yy.c. eg $ lex filename.l $ cc lex.yy.c
7. Compile that file with C compiler and verify the output.

INPUT

```c
#include <stdio.h>
#include <ctype.h>
#include <string.h>

int isKeyword(char buffer[]) {
    char keywords[5][10] = {"if", "else", "while", "for", "int"};
    for (int i = 0; i < 5; ++i) {
        if (strcmp(buffer, keywords[i]) == 0) {
            return 1;
        }
    }
    return 0;
}

int main() {
    char ch, buffer[15];
    char operators[] = "+-*/=";
    int i = 0;
```

```c
    printf("Enter your input: ");

    while ((ch = getchar()) != EOF) {
        if (strchr(operators, ch)) {
            printf("Operator: %c\n", ch);
        } else if (isalnum(ch)) {
            buffer[i++] = ch;
        } else if ((ch == ' ' || ch == '\n' || ch == '\t') && i != 0) {
            buffer[i] = '\0';

            if (isKeyword(buffer)) {
                printf("Keyword: %s\n", buffer);
            } else if (isdigit(buffer[0])) {
                printf("Number: %s\n", buffer);
            } else {
                printf("Identifier: %s\n", buffer);
            }
            i = 0;
        }
    }

    return 0;
}
```

OUTPUT

```
Enter your input: if (a == 10) { b = b + 5; }
Keyword: if
Identifier: a
Operator: =
Operator: =
Number: 10
Identifier: b
Operator: =
Identifier: b
Operator: +
Number: 5
```

RESULT

The lexical analyzer is implemented using lex and the output is verified.