

Design a lexical Analyzer to validate operators to recognize the operators +,-,\*,/ using regular Arithmetic operators .

**PGM:**

```
#include <stdio.h>
```

```
#include <string.h>
```

```
#define MAX_LENGTH 100
```

```
void checkOperators(const char *input) {
```

```
    int length = strlen(input);
```

```
    int i;
```

```
    printf("Identified operators:\n");
```

```
    for (i = 0; i < length; i++) {
```

```
        if (input[i] == '+' || input[i] == '-' || input[i] == '*' || input[i] == '/') {
```

```
            printf("Operator found: %c\n", input[i]);
```

```
        }
```

```
    }
```

```
}
```

```
int main() {
```

```
    char input[MAX_LENGTH];
```

```
    printf("Enter an arithmetic expression: ");
```

```
    fgets(input, MAX_LENGTH, stdin);
```

```
    // Remove newline character from the input if present
```

```
    input[strcspn(input, "\n")] = 0;
```

```
checkOperators(input);
```

```
return 0;
```

```
}
```

## OUTPUT:

main.c	Output
<pre>8  int i; 9 10 printf("Identified operators:\n"); 11 12 for (i = 0; i &lt; length; i++) { 13     if (input[i] == '+'    input[i] == '-'    input[i] == '*'            input[i] == '/') { 14         printf("Operator found: %c\n", input[i]); 15     } 16 } 17 } 18 19 int main() { 20     char input[MAX_LENGTH]; 21 22     printf("Enter an arithmetic expression: "); 23     fgets(input, MAX_LENGTH, stdin); 24 25     // Remove newline character from the input if present 26     input[strcspn(input, "\n")] = 0; 27 28     checkOperators(input); 29 30     return 0; 31 }</pre>	<pre>Enter an arithmetic expression: a+b/c Identified operators: Operator found: + Operator found: /  === Code Execution Successful ===</pre>