

**NAME:S Venkata Sai Naveen Kumar (192425176)**

**COURSE NAME : COMPILER DESIGN FOR CODE ANALYSIS**

**COURSE CODE : CSA1423**

**6. The lexical analyzer should ignore redundant spaces, tabs and new lines. It should also ignore comments. Although the syntax specification states that identifiers can be arbitrarily long, you may restrict the length to some reasonable value. Develop a lexical**

**Analyzer to identify identifiers, constants, operators using C program.**

C PROGRAMMING CODE:

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#define MAX\_LEN 31

int isOperator(char c) {

return (c == '+' || c == '-' || c == '\*' || c == '/' || c == '=' || c == '<' || c == '>' || c == '!' || c == '%' || c == '&' || c == '|' || c == '^'); } int main() { char c, buffer[100];

int i;

printf("Enter source code (end input with Ctrl+Z in Windows or Ctrl+D in Linux):\n"); while ((c = getchar()) != EOF) {

if (c == ' ' || c == '\t' || c == '\n') continue; if (c == '/') { char next = getchar(); if (next == '/') { while ((c = getchar()) != '\n' && c != EOF); continue;

} else if (next == '\*') { while ((c = getchar()) != EOF) { if (c == '\*' && (next = getchar()) == '/') break;

} continue; } else { ungetc(next, stdin); printf("Operator: %c\n", c); continue;

} } if (isalpha(c) || c == '\_') { i = 0; buffer[i++] = c; while ((c = getchar()) != EOF && (isalnum(c) || c == '\_')) { if (i < MAX\_LEN) buffer[i++] = c;

} buffer[i] = '\0'; printf("Identifier: %s\n", buffer); if (c != EOF) ungetc(c, stdin);

} else if (isdigit(c)) { i = 0; buffer[i++] = c; while ((c = getchar()) != EOF && isdigit(c)) { buffer[i++] = c;

} buffer[i] = '\0'; printf("Constant: %s\n", buffer); if (c != EOF) ungetc(c, stdin);

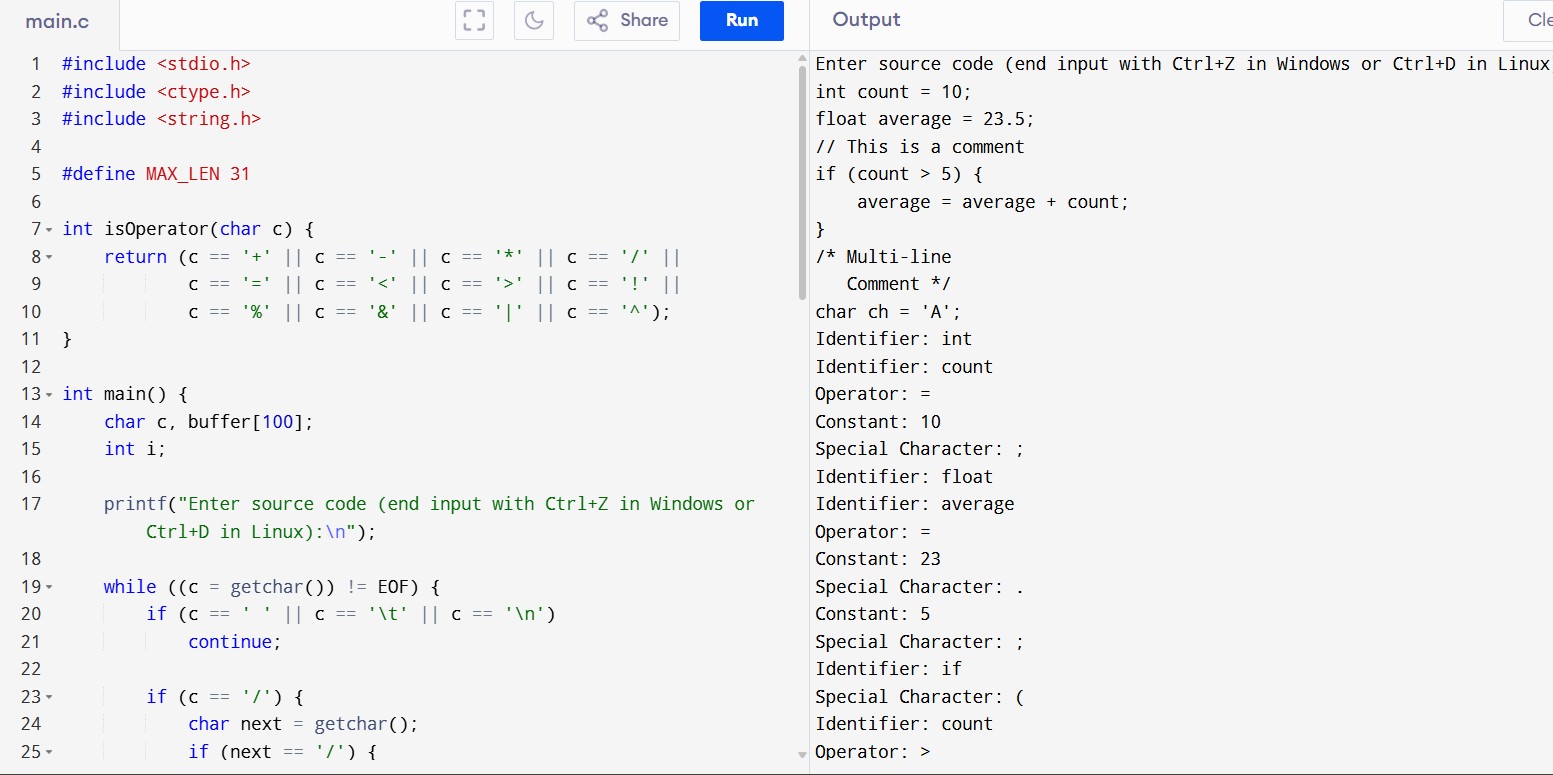
} else if (isOperator(c)) { printf("Operator: %c\n", c);

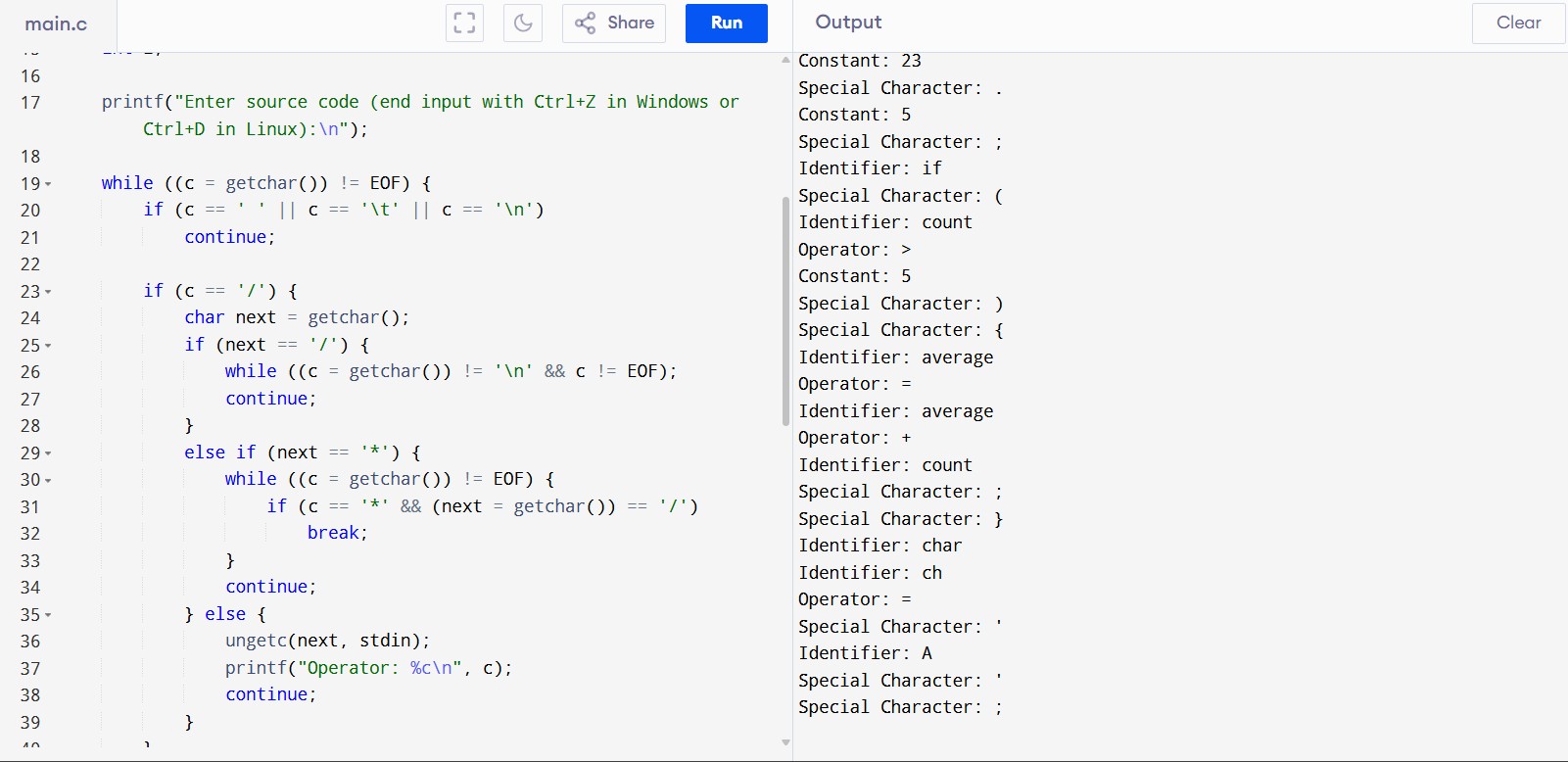
} else { printf("Special Character: %c\n", c);

} } return 0;

}

**OUTPUT:**





**7. Extend the lexical Analyzer to Check comments, dened as follows in C code:**

1. **A comment begins with // and includes all characters until the end of that line.**
2. **A comment begins with /\* and includes all characters through the next occurrence of the character sequence**

**\*/Develop a lexical Analyzer to identify whether a given line is a comment or not.**

C PROGRAMMING CODE:

#include <stdio.h>

#include <string.h>

#include <stdbool.h> #define MAX 200 int main() { char line[MAX]; bool inComment = false; printf("Enter code lines (Ctrl+D to end):\n");

while (fgets(line, MAX, stdin)) {

line[strcspn(line, "\n")] = '\0'; // Remove newline

if (inComment) {

printf("Comment Line: %s\n", line); if (strstr(line, "\*/")) { inComment = false;

}

} else {

if (strstr(line, "//") == line) { printf("Comment Line: %s\n", line);

}

else if (strstr(line, "/\*") == line) { printf("Comment Line: %s\n", line); if (!strstr(line, "\*/")) { inComment = true;

} } else {

printf("Not a Comment Line: %s\n", line);

}

}

}

return 0;

}

**OUTPUT:**

