**CSA1443-COMPILER DESIGN FOR INTRAPROCEDURAL ANALYSIS**

**NAME: KAUSHIK NARAYANAN V**

**REG NO: 192321047**

**PROGRAM 3**

**Design a lexical Analyzer for given language should ignore the redundant spaces, tabs and new lines and ignore comments using C**

**Aim**:  
To count the number of whitespace (spaces, tabs) and newline characters (\n) in a given input.

**Code:**

#include <stdio.h>

#include <ctype.h>

void skipWhitespaceAndComments(FILE \*fp) {

char ch;

while ((ch = fgetc(fp)) != EOF) {

if (isspace(ch)) continue; // Skip spaces, tabs, and newlines

if (ch == '/' && fgetc(fp) == '/') { // Skip single-line comment

while ((ch = fgetc(fp)) != '\n' && ch != EOF);

}

else if (ch == '/' && fgetc(fp) == '\*') { // Skip multi-line comment

while ((ch = fgetc(fp)) != '\*' || fgetc(fp) != '/')

if (ch == EOF) break;

} else {

ungetc(ch, fp); // Valid character to process

break;

}

}

}

void handleIdentifier(FILE \*fp) {

char token[100];

int index = 0;

char ch;

while (isalpha(ch = fgetc(fp)) || ch == '\_') token[index++] = ch;

token[index] = '\0';

printf("Identifier: %s\n", token);

ungetc(ch, fp);

}

void handleConstant(FILE \*fp) {

char token[100];

int index = 0;

char ch;

while (isdigit(ch = fgetc(fp))) token[index++] = ch;

token[index] = '\0';

printf("Constant: %s\n", token);

ungetc(ch, fp);

}

void handleOperator(char ch) {

printf("Operator: %c\n", ch);

}

void lexicalAnalyzer(FILE \*fp) {

char ch;

while ((ch = fgetc(fp)) != EOF) {

if (isspace(ch)) continue;

if (isalpha(ch) || ch == '\_') { ungetc(ch, fp); handleIdentifier(fp); }

else if (isdigit(ch)) { ungetc(ch, fp); handleConstant(fp); }

else if (ch == '+' || ch == '-' || ch == '\*' || ch == '/') handleOperator(ch);

else printf("Unrecognized character: %c\n", ch);

skipWhitespaceAndComments(fp); // Skip spaces and comments before next token

}

}

int main() {

FILE \*fp = fopen("source\_code.txt", "r");

if (!fp) { printf("File not found!\n"); return 1; }

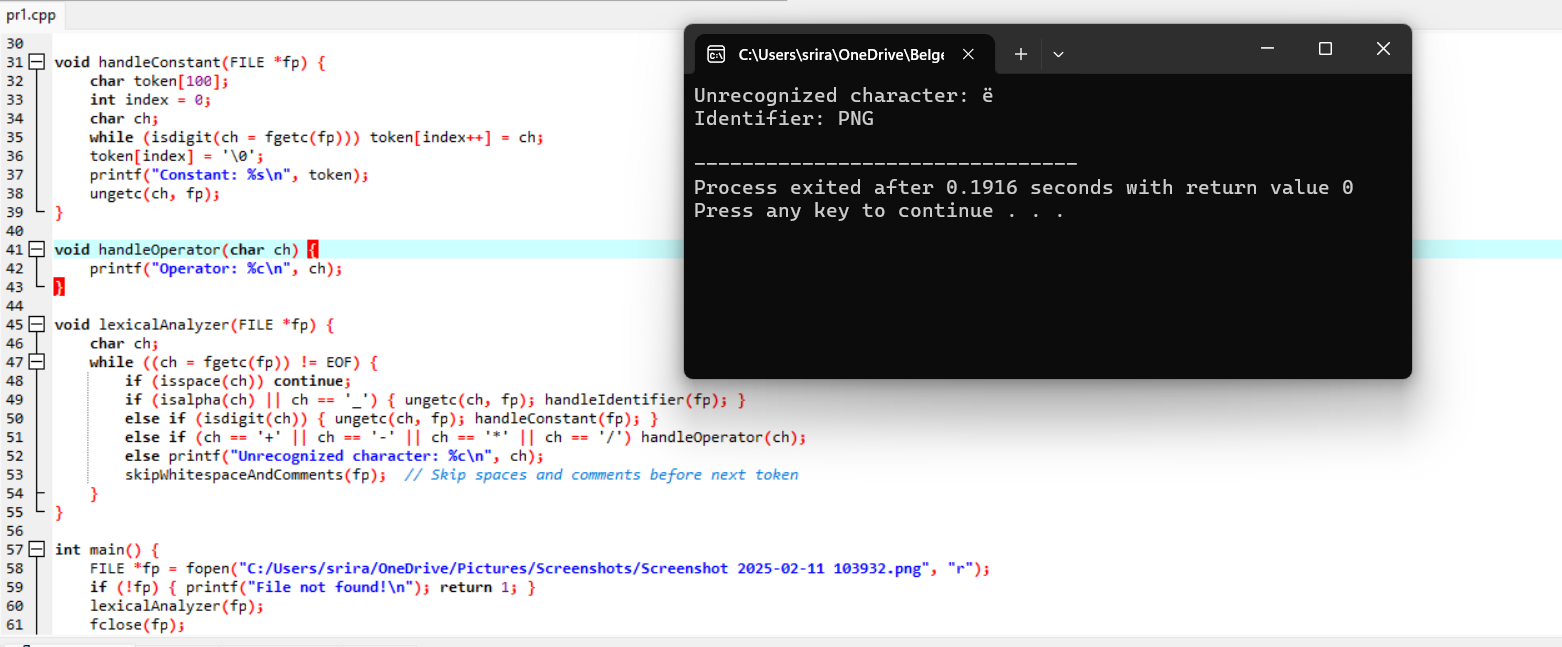
lexicalAnalyzer(fp);

fclose(fp);

return 0;

}

**Output:**

****