**Compiler Desigin program 3**

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

**Program:**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

int isIdentifier(char \*str) {

if (isalpha(str[0]) || str[0] == '\_') {

for (int i = 1; str[i] != '\0'; i++) {

if (!(isalnum(str[i]) || str[i] == '\_')) {

return 0;

}

}

return 1;

}

return 0;

}

int isConstant(char \*str) {

int len = strlen(str);

int dotCount = 0;

if (len == 0) return 0;

for (int i = 0; i < len; i++) {

if (str[i] == '.') {

dotCount++;

} else if (!isdigit(str[i])) {

return 0;

}

}

return (dotCount <= 1);

}

int isOperator(char ch) {

return (ch == '+' || ch == '-' || ch == '\*' || ch == '/' || ch == '=');

}

void skipComments(FILE \*sourceCode) {

char ch;

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

if (ch == '/') {

ch = fgetc(sourceCode);

if (ch == '/') {

while (ch != '\n' && ch != EOF) {

ch = fgetc(sourceCode);

}

} else if (ch == '\*') {

ch = fgetc(sourceCode);

while (ch != EOF) {

if (ch == '\*' && (ch = fgetc(sourceCode)) == '/') {

break;

ch = fgetc(sourceCode);

}

}

}

}

if (ch == EOF) break;

if (!isspace(ch)) {

ungetc(ch, sourceCode);

break;

}

}

}

void lexicalAnalyzer(FILE \*sourceCode) {

char token[100];

int tokenIndex = 0;

char ch;

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

if (isspace(ch)) {

continue;

}

if (ch == '/') {

skipComments(sourceCode);

continue;

}

if (isalnum(ch) || ch == '.') {

token[tokenIndex++] = ch;

while ((ch = fgetc(sourceCode)) != EOF && (isalnum(ch) || ch == '.')) {

token[tokenIndex++] = ch;

}

token[tokenIndex] = '\0';

ungetc(ch, sourceCode);

if (isIdentifier(token)) {

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

} else if (isConstant(token)) {

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

}

tokenIndex = 0;

} else if (isOperator(ch)) {

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

} else {

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

}

}

}

int main() {

FILE \*sourceCode;

char filename[100];

printf("Enter the filename of the source code: ");

scanf("%s", filename);

sourceCode = fopen(filename, "r");

if (sourceCode == NULL) {

printf("Error opening file.\n");

return -1;

}

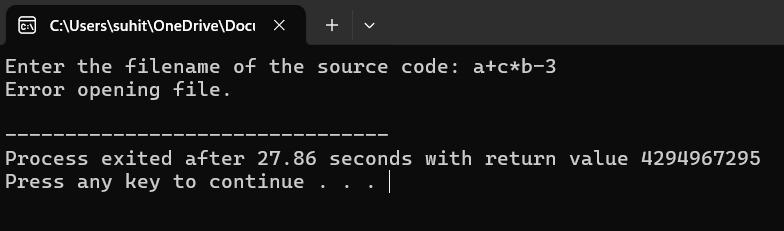
lexicalAnalyzer(sourceCode);

fclose(sourceCode);

return 0;

}

**Output:**

****