

**Compiler Design Lab**

**SUBMITTED BY: SUBMITTED TO:**

Name: Harsh Agrawal Dr. Ajay Prasad

SAP ID: 500107136

Enrollment No: R2142220326

Course- B.Tech. (CSE)

Batch: 4 (AI/ML) (Non- Hons)

Assignment-3

1. WAP to count number of ‘*scanf*’ and ‘*printf*’ statements in a C program. Replace them with ‘*read’* and ‘*write’* statements respectively.

Code-#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void replaceScanPrint(const char \*filename) {

FILE \*file = fopen(filename, "r");

if (!file) {

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

return;

}

FILE \*temp = fopen("output.c", "w");

if (!temp) {

printf("Error creating output file.\n");

fclose(file);

return;

}

char line[256];

int scanfCount = 0, printfCount = 0;

while (fgets(line, sizeof(line), file)) {

char \*pos;

if ((pos = strstr(line, "scanf")) != NULL) {

scanfCount++;

fprintf(temp, "%.\*sread%s", (int)(pos - line), line, pos + 5);

} else if ((pos = strstr(line, "printf")) != NULL) {

printfCount++;

fprintf(temp, "%.\*swrite%s", (int)(pos - line), line, pos + 6);

} else {

fputs(line, temp);

}

}

fclose(file);

fclose(temp);

printf("Number of scanf: %d\n", scanfCount);

printf("Number of printf: %d\n", printfCount);

}

int main() {

char filename[100];

printf("Enter filename: ");

scanf("%s", filename);

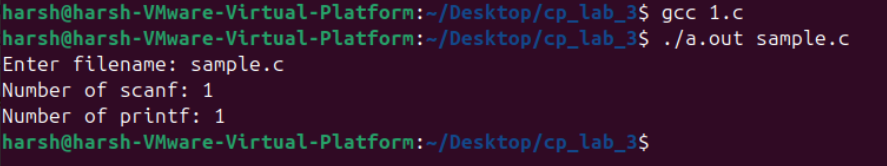
replaceScanPrint(filename);

return 0;

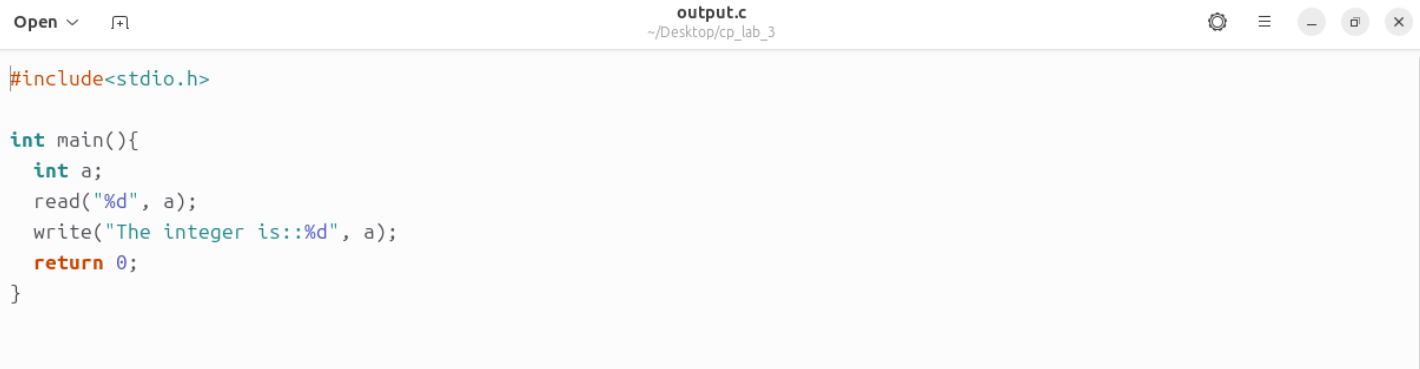
}

Sample.c file->

Output->



Output.c file after program can printf with write and scanf with read->



1. WAP to recognize and count the number of identifiers in a given input file.

Code->

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#define MAX\_WORD\_LEN 50

// List of C keywords

char \*keywords[] = {

"int", "float", "if", "else", "while", "return", "for", "char", "double",

"void", "switch", "case", "do", "break", "continue", "default", "extern",

"register", "static", "typedef", "union", "struct", "enum", "signed",

"unsigned", "const", "volatile", "short", "long", "sizeof", "goto"

};

int numKeywords = sizeof(keywords) / sizeof(keywords[0]);

// Function to check if a word is a keyword

int isKeyword(char \*word) {

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

if (strcmp(word, keywords[i]) == 0)

return 1;

}

return 0;

}

// Function to check if a word is a valid identifier

int isValidIdentifier(char \*word) {

if (!isalpha(word[0]) && word[0] != '\_') // First character must be a letter or ''

return 0;

return 1;

}

// Function to count identifiers in a file

void countIdentifiers(char \*filename) {

FILE \*file = fopen(filename, "r");

if (!file) {

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

return;

}

char word[MAX\_WORD\_LEN];

int identifierCount = 0;

char ch;

int index = 0;

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

if (isalnum(ch) || ch == '\_') { // If part of an identifier

word[index++] = ch;

} else {

if (index > 0) { // End of a word

word[index] = '\0'; // Null-terminate the string

index = 0;

if (!isKeyword(word) && isValidIdentifier(word)) {

identifierCount++;

printf("%s\n", word);

}

}

}

}

fclose(file);

printf("Total Identifiers: %d\n", identifierCount);

}

int main() {

char filename[100];

printf("Enter filename: ");

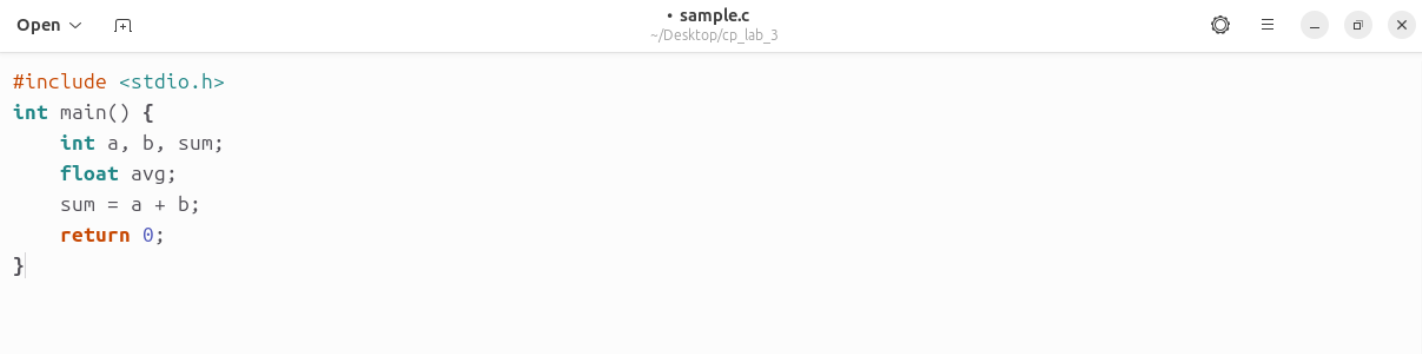
scanf("%s", filename);

countIdentifiers(filename);

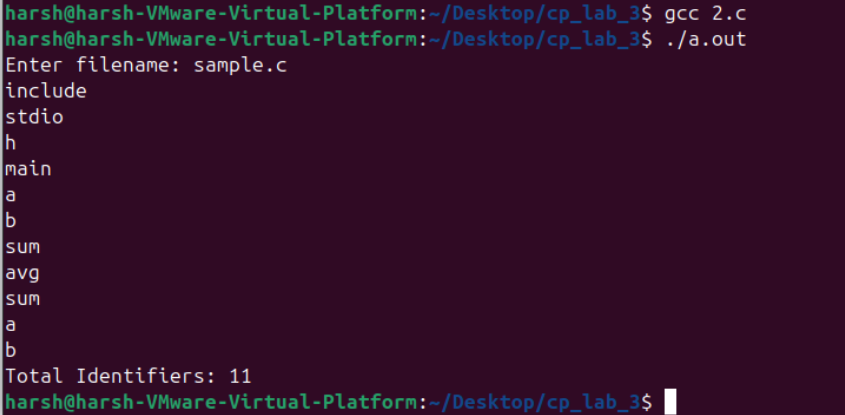
return 0;

}

Sample.c ->



Output->



1. WAP to recognize a valid arithmetic expression and identify the identifiers and operators present. Print them separately.

Code->

#include <stdio.h>

#include <ctype.h>

#include <string.h>

int isOperator(char ch) {

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

}

int isValidIdentifier(char \*word) {

if (!isalpha(word[0]) && word[0] != '\_')

return 0;

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

if (!isalnum(word[i]) && word[i] != '\_')

return 0;

}

return 1;

}

void analyzeExpression(char \*expression) {

char identifiers[10][20], operators[10];

int idCount = 0, opCount = 0;

char token[20];

int tokenIndex = 0;

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

if (isOperator(expression[i])) {

operators[opCount++] = expression[i];

} else if (isalnum(expression[i]) || expression[i] == '\_') {

token[tokenIndex++] = expression[i];

} else if (expression[i] == ' ' || expression[i] == ';') {

if (tokenIndex > 0) {

token[tokenIndex] = '\0';

if (isValidIdentifier(token)) {

strcpy(identifiers[idCount++], token);

}

tokenIndex = 0;

}

}

}

printf("Identifiers: ");

for (int i = 0; i < idCount; i++)

printf("%s ", identifiers[i]);

printf("\nOperators: ");

for (int i = 0; i < opCount; i++)

printf("%c ", operators[i]);

printf("\n");

}

int main() {

char expression[100];

printf("Enter an arithmetic expression: ");

fgets(expression, sizeof(expression), stdin);

analyzeExpression(expression);

return 0;

}

Output-> 