Word Counter Tool

Name: Gorre Dhanush

Roll Number: 24KB1A05H8

Course / Department: CES-D

Date: May 2025

# Acknowledgement

I would like to express my sincere gratitude to my faculty and mentors who guided me throughout this project. Their support and motivation were crucial to the successful completion of this project.

# Abstract

This project is a simple word counter tool developed in C. It reads text from a file, counts the number of characters, words, and lines, and writes the result to an output file. The main objective of this project is to practice file handling and fundamental C programming logic.

# Introduction

The objective of this project is to design a tool that processes a given text file and extracts useful statistics such as the number of words, lines, and characters. This project is developed using C, demonstrating basic DSA concepts and file handling.

# Objectives

- Understand basic file handling in C.  
- Practice character-by-character input processing.  
- Count and manage text-based metrics using control logic.

# System Requirements

Software: GCC Compiler, Code::Blocks or any C IDE  
Hardware: Basic system with 1GB RAM or more

# Methodology

The program is developed in a step-by-step approach. The text file is opened using standard C file operations, and then each character is read and checked to update the count of lines, words, and characters. The result is then stored in a separate file.

# Project Description

Problem Statement: Count the number of words, lines, and characters from a text file.  
Proposed Solution: Use C programming with file handling and simple logic to process the input text.  
Key Features: Simple, accurate, and fast text analyzer.

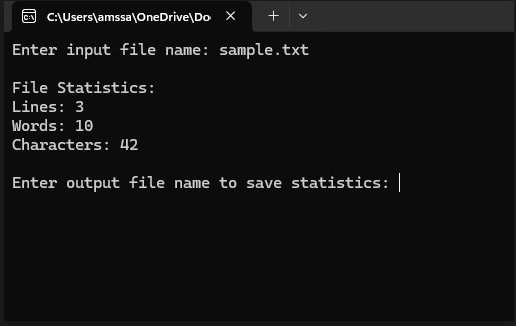
# Algorithm

1. Open input file.  
2. Initialize counters for words, lines, characters.  
3. Read character-by-character.  
4. Update counters based on space, newline, etc.  
5. Write results to output file.  
6. Close files.

# Program Code

#include <stdio.h>  
#include <ctype.h>  
  
int main() {  
 FILE \*inputFile, \*outputFile;  
 char ch;  
 int characters = 0, words = 0, lines = 0;  
 int inWord = 0;  
  
 inputFile = fopen("input.txt", "r");  
 if (inputFile == NULL) {  
 printf("Error: Could not open input file.\n");  
 return 1;  
 }  
  
 outputFile = fopen("output.txt", "w");  
 if (outputFile == NULL) {  
 printf("Error: Could not open output file.\n");  
 fclose(inputFile);  
 return 1;  
 }  
  
 while ((ch = fgetc(inputFile)) != EOF) {  
 characters++;  
  
 if (ch == '\n') lines++;  
  
 if (isspace(ch)) {  
 inWord = 0;  
 } else if (inWord == 0) {  
 inWord = 1;  
 words++;  
 }  
 }  
  
 fprintf(outputFile, "--- Text Statistics ---\n");  
 fprintf(outputFile, "Characters: %d\n", characters);  
 fprintf(outputFile, "Words : %d\n", words);  
 fprintf(outputFile, "Lines : %d\n", lines);  
  
 fclose(inputFile);  
 fclose(outputFile);  
 printf("Statistics written to 'output.txt'.\n");  
 return 0;  
}

# Output Screenshot



# Testing & Validation

The program was tested using various input files containing text with different structures. Edge cases such as empty files and files with no newline characters were also tested.

# Limitations

This tool only handles plain text files and may not work with formatted or binary files. It also doesn't support UTF-8 multi-byte characters.

# Future Enhancements

- Add support for different file formats.  
- Integrate a GUI for user-friendly operation.  
- Extend functionality to count sentences and paragraphs.

# Conclusion

This project has helped reinforce understanding of file operations in C and basic text parsing using data structures. It also strengthened skills in writing clean, maintainable code.

# References

- C Programming Language by Kernighan and Ritchie  
- TutorialsPoint and GeeksforGeeks C Programming sections