@mainpage

# Лабораторна робота № 12

### 1.1 Розробник

@author Іовов Микита ( КН-923Б) @date 20.05.2024 @version 1.8.17

### 1.2 Загальне завдання

Сформувати частотну таблицю символів у тексту та вивести її на екран (з вказанням кількості та процентного відношення). Обмеження - виводити тільки ті символи, що зустрічаються у тексті.

## 2 Опис програми

### 2.1 Функціональне призначення

Підрахунок частоти символів.

### 2.2 Обмеження на застосування

Обмеження - виводити тільки ті символи, що зустрічаються у тексті.

### 2.3 Опис логічної структури

* main.c: Основний файл програми, де ініціалізуються функції calculateFrequency() і print\_frequency\_table() для зчитування данних та підрахунку частотності символів:

#include <stdio.h>  
#include "lib.h"  
  
  
#define BUFFER\_SIZE 1000  
  
  
  
int main() {  
 char buffer[BUFFER\_SIZE];  
 int char\_count[ASCII\_SIZE] = {0};  
 int total\_chars = 0;  
  
 // Зчитуємо дані зі стандартного вводу  
 if (fgets(buffer, BUFFER\_SIZE, stdin) != NULL) {  
 // Підраховуємо кількість кожного символу  
 calculate\_frequency(buffer, char\_count, &total\_chars);  
  
 // Виводимо таблицю частот символів  
 print\_frequency\_table(char\_count, total\_chars);  
 } else {  
 fprintf(stderr, "Error reading input.\n");  
 return 1;  
 }  
  
 return 0;  
}

* lib.с: Файл з функціями calculateFrequency() і print\_frequency\_table() для зчитування данних та підрахунку частотності символів:

#include <stdio.h>  
#include "lib.h"  
  
void print\_frequency\_table(const int char\_count[ASCII\_SIZE], int total\_chars) {  
 printf("Character Frequency Table:\n");  
 printf("Char | Count | Percentage\n");  
 printf("-----+-------+-----------\n");  
  
 for (int i = 0; i < ASCII\_SIZE; i++) {  
 if (char\_count[i] > 0) {  
 double percentage = (char\_count[i] / (double)total\_chars) \* 100;  
 printf(" %c | %4d | %7.2f%%\n", i, char\_count[i], percentage);  
 }  
 }  
}  
  
  
void calculate\_frequency(const char \*buffer, int char\_count[ASCII\_SIZE], int \*total\_chars) {  
 \*total\_chars = 0;  
 for (size\_t i = 0; i < strlen(buffer); i++) {  
 char\_count[(unsigned char)buffer[i]]++;  
 (\*total\_chars)++;  
 }  
}

* lib.h: Заголовочний файл, що містить прототипи функцій calculateFrequency() та print\_frequency\_table:

/\*\*  
 \* @file lib.h  
 \* @brief Оголошення функції для підрахунку частоти символів у файлі.  
 \*/  
  
  
  
#ifndef LIB\_H  
#define LIB\_H  
  
#include <stdio.h>  
#include <stdlib.h>  
#include <ctype.h>  
#include <string.h>  
#define ASCII\_SIZE 256  
  
  
  
/\*\*  
 \* @brief Виводить таблицю частот символів.  
 \*   
 \* @details Ця функція приймає масив частот символів та загальну кількість символів,  
 \* потім виводить таблицю, яка показує частоту та відсоток кожного символу.  
 \*   
 \* @param char\_count Масив, що містить частоту кожного символу.  
 \* @param total\_chars Загальна кількість символів у вхідному буфері.  
 \*/  
  
void print\_frequency\_table(const int char\_count[ASCII\_SIZE], int total\_chars);  
  
  
  
  
/\*\*  
 \* @brief Розраховує частоту кожного символу в заданому буфері.  
 \*   
 \* @details Ця функція приймає буфер і розраховує частоту кожного символу  
 \* в буфері. Результати зберігаються в масиві char\_count, а загальна  
 \* кількість символів зберігається в total\_chars.  
 \*   
 \* @param buffer Вхідний рядок буфера.  
 \* @param char\_count Масив для зберігання частоти кожного символу.  
 \* @param total\_chars Вказівник на ціле число, в якому буде зберігатись загальна кількість символів.  
 \*/  
  
void calculate\_frequency(const char \*buffer, int char\_count[ASCII\_SIZE], int \*total\_chars);  
  
  
#endif

* test.c: Містить тестовий запуск функцій для завідомо відомих даних:

#include <check.h>  
#include <assert.h>  
#include "lib.h"  
  
  
  
  
/\*\*  
 \* @brief Тестовий випадок для функції calculate\_frequency.  
 \*   
 \* Цей тестовий випадок перевіряє правильність функції calculate\_frequency шляхом  
 \* порівняння розрахованих частот та загальної кількості символів з очікуваними значеннями.  
 \*/  
   
   
START\_TEST(test\_calculate\_frequency) {  
 char buffer[] = "hello world";  
 int char\_count[ASCII\_SIZE] = {0};  
 int total\_chars = 0;  
  
 calculate\_frequency(buffer, char\_count, &total\_chars);  
  
 ck\_assert\_int\_eq(char\_count['h'], 1);  
 ck\_assert\_int\_eq(char\_count['e'], 1);  
 ck\_assert\_int\_eq(char\_count['l'], 3);  
 ck\_assert\_int\_eq(char\_count['o'], 2);  
 ck\_assert\_int\_eq(char\_count[' '], 1);  
 ck\_assert\_int\_eq(char\_count['w'], 1);  
 ck\_assert\_int\_eq(char\_count['r'], 1);  
 ck\_assert\_int\_eq(char\_count['d'], 1);  
 ck\_assert\_int\_eq(total\_chars, 11);  
}  
END\_TEST  
  
  
  
/\*\*  
 \* @brief Створює тестовий набір для функцій роботи з частотою.  
 \*   
 \* Ця функція створює і повертає тестовий набір для функцій роботи з частотою.  
 \*   
 \* @return Suite\* Вказівник на створений тестовий набір.  
 \*/  
   
Suite \* frequency\_suite(void) {  
 Suite \*s;  
 TCase \*tc\_core;  
  
 s = suite\_create("Frequency");  
  
 /\* Core test case \*/  
 tc\_core = tcase\_create("Core");  
  
 tcase\_add\_test(tc\_core, test\_calculate\_frequency);  
 suite\_add\_tcase(s, tc\_core);  
  
 return s;  
}  
  
  
/\*\*  
 \* @brief Основна функція для запуску тестового набору.  
 \*   
 \* Ця функція запускає створений тестовий набір і повертає результат.  
 \*   
 \* @return int Повертає 0, якщо всі тести пройшли успішно, 1, якщо будь-який тест провалився.  
 \*/  
   
   
   
int main(void) {  
 int number\_failed;  
 Suite \*s;  
 SRunner \*sr;  
  
 s = frequency\_suite();  
 sr = srunner\_create(s);  
  
 srunner\_run\_all(sr, CK\_NORMAL);  
 number\_failed = srunner\_ntests\_failed(sr);  
 srunner\_free(sr);  
 return (number\_failed == 0) ? 0 : 1;  
}

### 2.2.3 Структура проекту

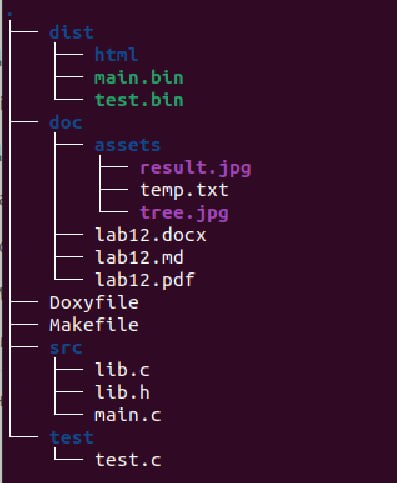


Рисунок 1 - Структура проекту

### 2.4 Важливі фрагменти програми

* Включення заголовочних файлів.
* Створення тестового набору.
* Запуск тестів та вивід результатів.

# 2 Варіанти використання

Для представлення виконання кожного завдання використовується: - виведення результатів у консоль за допомогою функції виводу; - юніт-тест;

### Варіант використання 1

* Формування таблиці частотності символів.

Інструкція для запуску програми за допомогою юніт-тесту: - Виклик юніт тесту за допомогою команди ./test.bin.

Інструкція для запуску програми: - Ввести команду: cat ./assets/input.txt | ./dist/main.bin для зчитування данних з текстового файлу програмою main.bin.

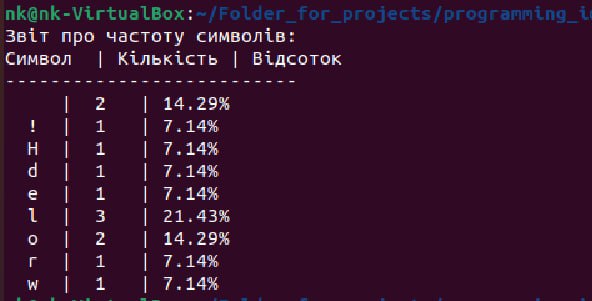


Рисунок 2 - Скріншот виводу програми

### Результат:

Як бачимо, програма коректно формує таблицю частотності символів текстового файла.

# Висновок

Програма є ефективним інструментом для статистичного аналізу текстових даних. Вона дозволяє отримати детальну інформацію про розподіл символів у тексті, що може бути корисно для різних дослідницьких та прикладних завдань. Використання стандартних бібліотек C дозволяє забезпечити коректність роботи з файлами та символами, а також спрощує реалізацію основних функцій програми.