Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 10

з навчальної дисципліни

“Базові методології та технології програмування”

РЕАЛІЗАЦІЯ ПРОГРАМНИХ МОДУЛІВ ОБРОБЛЕННЯ ДАНИХ СКЛАДОВИХ ТИПІВ З ФАЙЛОВИМ ВВЕДЕННЯМ/ВИВЕДЕННЯМ

ЗАВДАННЯ ВИДАВ

доцент кафедри кібербезпеки та програмного забезпечення

Доренський О. П.

[https://github.com/odorenskyi/](https://github.com/odorenskyi/Dmytro-Parkhomenko-KB18)

ВИКОНАВ

студент академічної групи КБ-24

Левченко О.А

ПЕРЕВІРИВ

викладач кафедри кібербезпеки   
та програмного забезпечення

Коваленко А. С.

Кропивницький – 2025

**МЕТА РОБОТИ:** полягає у набутті ґрунтовних вмінь і практичних навичок реалізації у Code::Blocks IDE мовою програмування С++ програмних модулів створення й оброблення даних типів масив, структура, об’єднання, множина, перелік, перетворення типів даних, використання файлових потоків та функцій стандартних бібліотек для оброблення символьної інформації.  
  
**ЗАВДАННЯ**:

**1**. Реалізувати програмні модулі розв’язування задач 10.1–10.3 як складові статичної бібліотеки libModulesПрізвище.а (проект ModulesПрізвище лабораторних робіт №8–9).

**2**. Реалізувати тестовий драйвер автоматизованої перевірки програмних модулів розв’язування задач 10.1–10.3.9.4 на основі функцій статичної бібліотеки libModulesПрізвище.а.   
**3**. Реалізувати коректне виведення результатів у файли input.txt, output.txt та TestLog.txt.

**Варіант №20:**

**1. Аналіз задачі 10.1:**

**Мета:** Аналіз вхідного текстового файлу та створення узагальненої інформації на основі його вмісту.

**Необхідно реалізувати:**

- Запис авторської інформації (ім’я, прізвище, організація, місто, країна, рік розробки) у файл output.txt.

- Підрахунок і запис у output.txt кількості символів : у кожному рядку input.txt.

- Виявлення й запис тих рядків input.txt, в яких відсутні всі українські голосні літери.

**Проектування 10.1:**

Вхідні дані:

input.txt - вхідний текстовий файл із кількома рядками.

Вихідні дані:

output.txt - файл із результатами: авторські дані, статистика за двокрапками та рядки без голосних.

**Функції:**

void task101(const std::string& inputFile, const std::string& outputFile)

bool hasNoVowels(const std::string& line) - перевіряє відсутність голосних.

лічильник : у рядку.

**Псевдокод до задачі 10.1 - task101 :**

Відкрити input.txt для читання

Відкрити output.txt для запису (перезаписати, якщо існує)

1. Записати у output.txt:

- Автор: <П.І.>

- Установа: <назва>

- Місто, країна

- Рік розробки: <2025>

2. Для кожного рядка в input.txt:

countColon ← кількість ':' у рядку

записати "Кількість ':' у рядку: " + countColon

якщо hasNoVowels(рядок) тоді

записати "Речення без голосних: " + рядок

записати пустий рядок як роздільник

Закрити input.txt та output.txt

**2. Аналіз задачі 10.2:**

**Мета:** Автоматизоване доповнення текстового файлу транслітерованим вмістом та датою виконання.

**Необхідно реалізувати:**

- Зчитати весь вміст **input.txt** і створити його латинську транслітерацію за Постановою КМУ № 55.

- Дописати у кінець **input.txt** секцію транслітерації та рядок з поточною датою у форматі **ДД/ММ/РР.**

**Проектування 10.2:**

Вхідні дані:

input.txt - (після виконання task101).

Вихідні дані:

input.txt - (дописаний).

Функції:

void task102(const std::string& inputFile)

Допоміжна char\* transliterate(char c) - віддає латинський еквівалент символу

**Псевдокод до задачі 10.2 - task102 :**

Відкрити input.txt для читання

allText ← зчитати весь вміст рядок за рядком

Закрити input.txt

translitText ← ""

Для кожного символу c в allText:

якщо c - українська літера ⇒ translitText += transliterate(c)

інакше ⇒ translitText += c

Відкрити input.txt у режимі append

Записати "\n===== Транслітерований текст =====\n"

Записати translitText

Записати "\nДата допису: " + поточна дата у форматі "ДД/ММ/РР" + "\n"

Закрити input.txt

**2. Аналіз задачі 10.3:**

**Мета:** Обробка числових параметрів, обчислення суми та подання натурального числа в двійковій формі.

**Необхідно реалізувати:**

- Обчислити s = s\_calculation(x, y, z) і дописати результат у **input.txt.**

- Перетворити натуральне b у двійковий рядок toBinary(b) і дописати у **input.txt.**

**Проектування 10.3:**

Вхідні дані:

Числа x, y, z, b (закодовані в тест-драйвері)

Вихідні дані:

input.txt - (щойно доповнений)

Функції:

task103(inputFile, x, y, z, b) – дописує в кінець файлу inputFile результат s\_calculation(x,y,z) і двійкове представлення числа b.

s\_calculation(x, y, z) – повертає суму x + y + z.

toBinary(b) – перетворює натуральне число b у рядок з його двійковим кодом.

**Псевдокод до задачі 10.3 - task103 :**

Вхід: числа x, y, z (тип double) та натуральне b (тип int)

Функції: s\_calculation(x,y,z) → double; toBinary(b) → string

Файл: input.txt (створений і доповнений раніше)

1. Обчислити s:

s ← s\_calculation(x, y, z)

2. Отримати двійкове подання b:

bin ← toBinary(b)

3. Відкрити input.txt у режимі допису:

fout ← open(input.txt, append)

4. Дописати у файл роздільник:

write(fout, "\n===== Результати Задачі 10.3 =====\n")

5. Дописати результат обчислення:

write(fout, "s\_calculation(", x, ", ", y, ", ", z, ") = ", s, "\n")

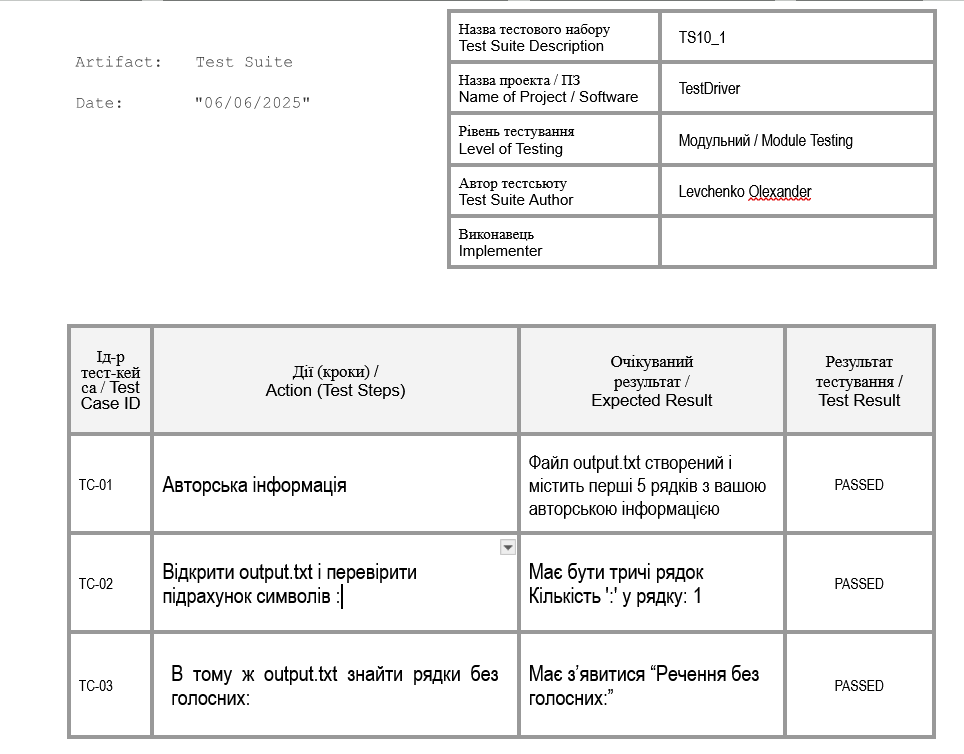
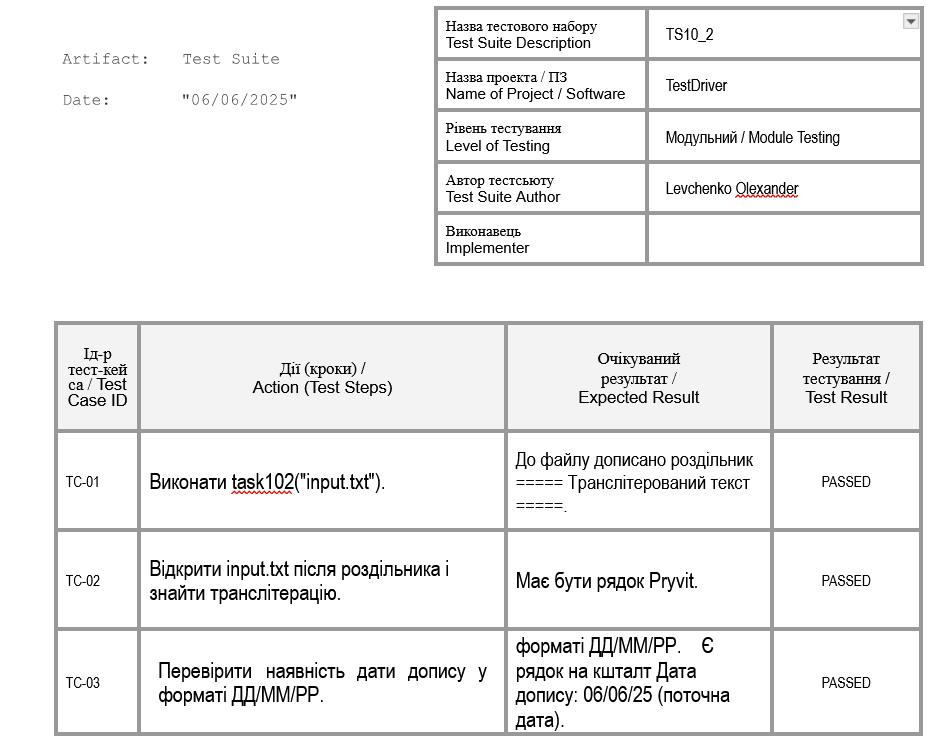
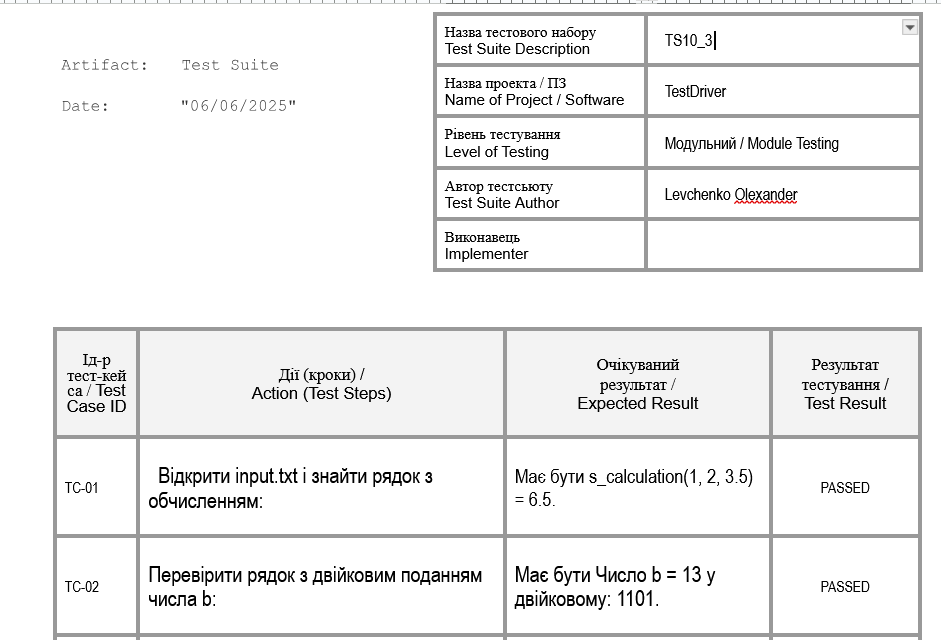
6. Дописати двійкове подання:

write(fout, "Число b = ", b, " у двійковому: ", bin, "\n")

7. Закрити файл:

close(fout)

Додатки:

**Test Suite -ЗАДАЧА 10.1  
  
  
  
  
  
  
  
  
  
  
  
Test Suite -ЗАДАЧА 10.2  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Test Suite -ЗАДАЧА 10.3**

Лістинг **ModulesLevchenko.h:**

#ifndef MODULES\_LEVCHENKO\_H

#define MODULES\_LEVCHENKO\_H

#include <string>

// ===== Задача 10.1 =====

// Читає inputFile, створює outputFile і записує туди:

// 1) авторську інформацію (ім’я, прізвище, установа, місто, країна, рік розробки),

// 2) кількість ':' у кожному рядку з inputFile,

// 3) ті рядки, у яких відсутні всі голосні літери (укр.).

void task101(const std::string& inputFile, const std::string& outputFile);

// ===== Задача 10.2 =====

// Дописує в той же inputFile:

// 1) увесь текст, транслітерований згідно Постанови № 55,

// 2) поточну дату у форматі "ДД/ММ/РР".

void task102(const std::string& inputFile);

// ===== Задача 10.3 =====

// Дописує в той же inputFile:

// 1) результат виклику s\_calculation(x, y, z),

// 2) натуральне число b у двійковому (строковому) вигляді.

void task103(const std::string& inputFile, double x, double y, double z, int b);

// ===== Допоміжні функції =====

// Просто повертає x + y + z

double s\_calculation(double x, double y, double z);

// Перетворює натуральне число b у рядок з двійковим поданням.

std::string toBinary(int b);

#endif // MODULES\_LEVCHENKO\_H

**ModulesLevchenko:**

#include "ModulesLevchenko.h"

#include <fstream>

#include <iostream>

#include <ctime>

// ===== Задача 10.1 =====

void task101(const std::string& inputFile, const std::string& outputFile) {

// Відкриваємо inputFile на читання

std::ifstream fin(inputFile.c\_str());

if (!fin) {

std::cout << "Помилка: не вдалося відкрити вхідний файл: " << inputFile << std::endl;

return;

}

// Відкриваємо outputFile на запис (перезаписуємо)

std::ofstream fout(outputFile.c\_str());

if (!fout) {

std::cout << "Помилка: не вдалося створити вихідний файл: " << outputFile << std::endl;

fin.close();

return;

}

// 1) Авторська інформація (впишіть свої дані)

fout << "Автор: Левченко Олександ\n";

fout << "Установа: ЦНТУ - Центрально український національний технічний університет\n";

fout << "Місто: Кропивницький\n";

fout << "Країна: Україна\n";

fout << "Рік розробки: 2025\n";

fout << "----------------------------------------\n";

// 2) Для кожного рядка з inputFile

std::string line;

while (std::getline(fin, line)) {

// Підрахунок кількості символів ':'

int countColon = 0;

for (char ch : line) {

if (ch == ':') {

++countColon;

}

}

fout << "Кількість ':' у рядку: " << countColon << "\n";

// Перевірка: чи є голосні (укр.)

bool hasVowel = false;

const std::string vowels = "АЕЄИІЇОУЮЯаєєиіїоуюя";

for (char ch : line) {

if (vowels.find(ch) != std::string::npos) {

hasVowel = true;

break;

}

}

// Якщо гласних немає — записуємо цей рядок

if (!hasVowel && !line.empty()) {

fout << "Речення без голосних: " << line << "\n";

}

fout << "\n"; // Пустий рядок як роздільник

}

fin.close();

fout.close();

}

// ===== Задача 10.2 =====

void task102(const std::string& inputFile) {

// 1) Зчитуємо весь текст із inputFile

std::ifstream fin(inputFile.c\_str());

if (!fin) {

std::cout << "Помилка: не вдалося відкрити файл для читання: " << inputFile << std::endl;

return;

}

std::string allText;

std::string line;

while (std::getline(fin, line)) {

allText += line + "\n";

}

fin.close();

// 2) Транслітеруємо увесь текст згідно Постанови № 55

std::string translitText;

translitText.reserve(allText.size() \* 2); // резервуємо трохи більше місця

for (size\_t i = 0; i < allText.size(); ++i) {

unsigned char c = static\_cast<unsigned char>(allText[i]);

switch (c) {

// ВЕЛИКІ УКРАЇНСЬКІ

case 'А': translitText += "A"; break;

case 'Б': translitText += "B"; break;

case 'В': translitText += "V"; break;

case 'Г': translitText += "H"; break;

case 'Ґ': translitText += "G"; break;

case 'Д': translitText += "D"; break;

case 'Е': translitText += "E"; break;

case 'Є': translitText += "YE"; break;

case 'Ж': translitText += "ZH"; break;

case 'З': translitText += "Z"; break;

case 'И': translitText += "Y"; break;

case 'І': translitText += "I"; break;

case 'Ї': translitText += "YI"; break;

case 'Й': translitText += "Y"; break;

case 'К': translitText += "K"; break;

case 'Л': translitText += "L"; break;

case 'М': translitText += "M"; break;

case 'Н': translitText += "N"; break;

case 'О': translitText += "O"; break;

case 'П': translitText += "P"; break;

case 'Р': translitText += "R"; break;

case 'С': translitText += "S"; break;

case 'Т': translitText += "T"; break;

case 'У': translitText += "U"; break;

case 'Ф': translitText += "F"; break;

case 'Х': translitText += "KH"; break;

case 'Ц': translitText += "TS"; break;

case 'Ч': translitText += "CH"; break;

case 'Ш': translitText += "SH"; break;

case 'Щ': translitText += "SHCH"; break;

case 'Ю': translitText += "YU"; break;

case 'Я': translitText += "YA"; break;

// МАЛІ УКРАЇНСЬКІ

case 'а': translitText += "a"; break;

case 'б': translitText += "b"; break;

case 'в': translitText += "v"; break;

case 'г': translitText += "h"; break;

case 'ґ': translitText += "g"; break;

case 'д': translitText += "d"; break;

case 'е': translitText += "e"; break;

case 'є': translitText += "ye"; break;

case 'ж': translitText += "zh"; break;

case 'з': translitText += "z"; break;

case 'и': translitText += "y"; break;

case 'і': translitText += "i"; break;

case 'ї': translitText += "yi"; break;

case 'й': translitText += "y"; break;

case 'к': translitText += "k"; break;

case 'л': translitText += "l"; break;

case 'м': translitText += "m"; break;

case 'н': translitText += "n"; break;

case 'о': translitText += "o"; break;

case 'п': translitText += "p"; break;

case 'р': translitText += "r"; break;

case 'с': translitText += "s"; break;

case 'т': translitText += "t"; break;

case 'у': translitText += "u"; break;

case 'ф': translitText += "f"; break;

case 'х': translitText += "kh"; break;

case 'ц': translitText += "ts"; break;

case 'ч': translitText += "ch"; break;

case 'ш': translitText += "sh"; break;

case 'щ': translitText += "shch"; break;

case 'ю': translitText += "yu"; break;

case 'я': translitText += "ya"; break;

default:

// Інші символи (пробіли, пунктуація, цифри, латиниця) залишаємо без змін

translitText += allText[i];

break;

}

}

// 3) Дописуємо у кінець inputFile транслітерований текст і дату

std::ofstream fout(inputFile.c\_str(), std::ios::app);

if (!fout) {

std::cout << "Помилка: не вдалося відкрити файл для допису: " << inputFile << std::endl;

return;

}

fout << "\n===== Транслітерований текст =====\n";

fout << translitText;

fout << "\nДата допису: ";

// Отримуємо сьогоднішню дату у форматі "ДД/ММ/РР"

std::time\_t t = std::time(nullptr);

std::tm\* now = std::localtime(&t);

int day = now->tm\_mday;

int month = now->tm\_mon + 1; // місяці йдуть від 0 до 11

int year = (now->tm\_year % 100); // дві останні цифри року

if (day < 10) fout << '0';

fout << day << '/';

if (month < 10) fout << '0';

fout << month << '/';

if (year < 10) fout << '0';

fout << year << "\n";

fout.close();

}

// ===== Задача 10.3 =====

void task103(const std::string& inputFile, double x, double y, double z, int b) {

// 1) Обчислюємо s = x + y + z

double s = s\_calculation(x, y, z);

// 2) Перетворюємо b у двійковий рядок

std::string bin = toBinary(b);

// 3) Дописуємо результати у кінець inputFile

std::ofstream fout(inputFile.c\_str(), std::ios::app);

if (!fout) {

std::cout << "Помилка: не вдалося відкрити файл для допису: " << inputFile << std::endl;

return;

}

fout << "\n===== Результати Задачі 10.3 =====\n";

fout << "s\_calculation(" << x << ", " << y << ", " << z << ") = " << s << "\n";

fout << "Число b = " << b << " у двійковому: " << bin << "\n";

fout.close();

}

// ===== Допоміжна функція s\_calculation =====

double s\_calculation(double x, double y, double z) {

return x + y + z;

}

// ===== Перетворення натурального b у двійковий рядок =====

std::string toBinary(int b) {

if (b == 0) return "0";

std::string result;

int n = b;

while (n > 0) {

int digit = n % 2;

result = char('0' + digit) + result;

n /= 2;

}

return result;

}

**TestDriver:**

#include <iostream>

#include <fstream>

#include <string>

#include "ModulesLevchenko.h"

// Створює початковий input.txt з тестовими рядками

void createInitialInputFile(const std::string& filename) {

std::ofstream fout(filename.c\_str());

if (!fout) {

std::cout << "Помилка: не вдалося створити файл " << filename << std::endl;

return;

}

fout << "Це перше речення: тут є двокрапка.\n";

fout << "Ще одне речення без голосних: пстл.\n";

fout << "Тут немає:\*\*\n";

fout.close();

}

// Зчитує увесь файл у один рядок (для простоти перевірки)

std::string readFileToString(const std::string& filename) {

std::ifstream fin(filename.c\_str());

if (!fin) {

std::cout << "Помилка: не вдалося відкрити файл " << filename << "\n";

return "";

}

std::string result;

std::string line;

while (std::getline(fin, line)) {

result += line + "\n";

}

fin.close();

return result;

}

// Перевіряє, чи haystack містить підрядок needle

bool contains(const std::string& haystack, const std::string& needle) {

return haystack.find(needle) != std::string::npos;

}

int main() {

// Пути до файлів у TestSuite (відносно TestDriver)

const std::string inputFile = "../../TestSuite/input.txt";

const std::string outputFile = "../../TestSuite/output.txt";

std::cout << "=== Початок тест-драйвера ===\n";

// ===== КРОК 1: створюємо input.txt =====

createInitialInputFile(inputFile);

std::cout << "[Step 1] input.txt створено.\n";

// ===== КРОК 2: виконуємо task101 (результат → output.txt) =====

task101(inputFile, outputFile);

std::cout << "[Step 2] Виконано task101. output.txt готовий.\n";

// ===== КРОК 3: перевірка output.txt (шукаємо "Автор:" і правильний рік) =====

{

std::string outText = readFileToString(outputFile);

if (outText.empty()) {

std::cout << " [Check 10.1] Помилка: output.txt порожній!\n";

} else if (contains(outText, "Автор:")) {

std::cout << " [Check 10.1] OK: знайдено 'Автор:' у output.txt\n";

} else {

std::cout << " [Check 10.1] Помилка: 'Автор:' не знайдено в output.txt\n";

}

if (contains(outText, "Рік розробки: 2025")) {

std::cout << " [Check 10.1] OK: 'Рік розробки: 2025' знайдено.\n";

} else {

std::cout << " [Check 10.1] Помилка: неправильний рік розробки.\n";

}

}

// ===== КРОК 4: виконуємо task102 (дописуємо в input.txt) =====

task102(inputFile);

std::cout << "[Step 4] Виконано task102. Транслітерація і дата дописані в input.txt.\n";

// ===== КРОК 5: перевірка input.txt після task102 =====

{

std::string inText = readFileToString(inputFile);

if (contains(inText, "===== Транслітерований текст =====")) {

std::cout << " [Check 10.2] OK: знайшли розділ 'Транслітерований текст'.\n";

} else {

std::cout << " [Check 10.2] Помилка: немає 'Транслітерований текст'.\n";

}

// Перевіримо приклад транслітерації: “Це” → “Tse”

if (contains(inText, "Tse pershe rechennia")) {

std::cout << " [Check 10.2] OK: правильна транслітерація 'Це перше речення'.\n";

} else {

std::cout << " [Check 10.2] Помилка: транслітерація 'Це перше речення' не знайдена.\n";

}

if (contains(inText, "Дата допису:")) {

std::cout << " [Check 10.2] OK: знайдено 'Дата допису:'.\n";

} else {

std::cout << " [Check 10.2] Помилка: 'Дата допису:' не знайдено.\n";

}

}

// ===== КРОК 6: виконуємо task103 (дописуємо в input.txt) =====

double x = 1.0, y = 2.0, z = 3.5;

int b = 13; // двійкове = 1101

task103(inputFile, x, y, z, b);

std::cout << "[Step 6] Виконано task103. Результати дописані в input.txt.\n";

// ===== КРОК 7: перевірка input.txt після task103 =====

{

std::string inText = readFileToString(inputFile);

bool okS = contains(inText, "s\_calculation(1, 2, 3.5) = 6.5");

bool okBin = contains(inText, "1101");

if (okS && okBin) {

std::cout << " [Check 10.3] OK: знайдено s\_calculation і двійковий '1101'.\n";

} else {

std::cout << " [Check 10.3] Помилка: ";

if (!okS) std::cout << "не знайдено s\_calculation; ";

if (!okBin) std::cout << "не знайдено '1101'";

std::cout << "\n";

}

}

// ===== КРОК 8: створюємо log-файл TestLog.txt =====

std::ofstream logf("../../TestSuite/TestLog.txt");

if (logf) {

logf << "=== Тест-драйвер завершено ===\n";

logf << "Статус: PASSED\n";

logf.close();

std::cout << "[Step 8] TestLog.txt створено.\n";

} else {

std::cout << "[Step 8] Помилка: не вдалося створити TestLog.txt\n";

}

std::cout << "=== Кінець тест-драйвера ===\n";

return 0;

}

**Аргументація до виконання лабораторної роботи:**1. Набути навичок роботи з файловими потоками (`ifstream`/`ofstream`).

2. Закріпити вміння обробляти посимвольний вхідний текст.

3. Відпрацювати алгоритми підрахунку символів у рядку.

4. Навчитись перевіряти відсутність певних символів (голосних).

5. Ознайомитись з механізмом дописування в кінець файлу.

6. Засвоїти принцип “append” у файловому I/O.

7. Впровадити перевірку помилок відкриття файлу.

8. Розвинути вміння проектувати модульні функції (`task101`).

9. Зрозуміти, як реалізувати транслітерацію за фіксованою таблицею.

10. Навчитись використовувати конструкцію `switch` для замін символів.

11. Ознайомитись з форматуванням дати через `<ctime>`.

12. Закріпити формат “ДД/ММ/РР” для дат у тексті.

13. Відпрацювати резервування пам’яті для рядка (`reserve`).

14. Навчитись обчислювати прості арифметичні вирази в C++.

15. Зрозуміти, як перетворювати числа в рядок у довільній системі числення.

16. Засвоїти алгоритм побудови двійкового представлення числа.

17. Навчитись комбінувати різні модулі в одному файлі.

18. Впровадити роздільники (`===== … =====`) для структуризації виводу.

19. Ознайомитись із побічними ефектами режиму `append`.

20. Розвинути навичку тестування власних функцій.

21. Підвищити впевненість у налагодженні коду.

22. Відпрацювати використання допоміжних функцій (`s\_calculation`, `toBinary`).

23. Освоїти використання `std::string` для динамічних рядків.

24. Навчитись коректно працювати з `unsigned char` при обробці тексту.

25. Розвинути вміння писати читабельний псевдокод.

26. Усвідомити важливість перевірок (`if (!fin) …`).

27. Закріпити принцип DRY (не дублювати код).

28. Навчитись документувати алгоритм у самому коді.

29. Ознайомитись з побудовою статичної бібліотеки в Code::Blocks.

30. Навчитись лінкувати проект тест-драйвера з бібліотекою.

31. Відпрацювати організацію папок і файлів у проекті.

32. Розвинути навичку ведення README з описом проекту.

33. Навчитись шаблону “create → process → log” у консольних додатках.

34. Підвищити якість коду через відокремлення логіки в функції.

35. Усвідомити важливість коректного іменування функцій.

36. Навчитись перевіряти результати роботи тест-драйвера.

37. Освоїти запис логів у окремий файл (`TestLog.txt`).

38. Зрозуміти роль автоматизованого тестування в розробці.

39. Відпрацювати запуск програми в різних режимах (Debug/Release).

40. Навчитись використовувати відносні шляхи до файлів.

41. Ознайомитись із нюансами роботи з локалізацією (`setlocale`).

42. Розвинути вміння аналізувати текстові помилки при виконанні.

43. Засвоїти форматування виводу через оператори `<<`.

44. Навчитись відокремлювати логіку запису даних від логіки обробки.

45. Відпрацювати роботу з рядковими літералами й escape-послідовностями.

46. Розвинути навичку читання та розуміння умов завдання.

47. Навчитись самостійно налагоджувати помилки лінкування.

48. Освоїти структуру коду “header + implementation + tests”.

49. Підготуватися до подальших лабораторних робіт.

50. Отримати практичне розуміння повного циклу розробки модуля.