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 ТИІВ 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

Бовкун Д. О.

ПЕРЕВІРИВ

викладач кафедри кібербезпеки   
та програмного забезпечення

Коваленко А. С.

Кропивницький – 2025

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

Мета роботи полягає у набутті грунтовних 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 libModules Прізвище. а (проект Modules Прізвище лабораторних робіт №8-9).
2. Реалізувати тестовий драйвер автоматизованої перевірки програмних модулів розв'язування задач 10.1-10.3.

Варіант № 17

Аналіз та постановка задач

Задача 10.1

Мета: Обробити вхідний текстовий файл та записати у вихідний файл певну інформацію.  
Вхідні дані: Текстовий файл з довільним текстом українською мовою.  
Вихідні дані: Текстовий файл, який містить:

1. Авторську інформацію (ім'я, прізвище, установа, місто, країна, рік розробки).
2. Кількість абзаців у вхідному тексті.
3. Перевірку наявності слів "Україна", "університет", "блокнот" (з урахуванням різних форм слів).

Кроки виконання:

1. Відкрити вхідний файл для читання та вихідний файл для запису (якщо він існує, перезаписати).
2. Записати авторську інформацію у вихідний файл.
3. Підрахувати кількість абзаців у вхідному файлі (абзаци можуть бути розділені порожніми рядками або іншими символами).
4. Перевірити наявність слів "Україна", "університет", "блокнот" у тексті, ігноруючи форми слів (наприклад, "університету").
5. Записати результати у вихідний файл.
6. Закрити файли.

Особливості:

Використання файлових потоків ifstream та ofstream.

Перевірка на коректність відкриття файлів.

Обробка текстових даних з урахуванням української мови.

Задача 10.2

Мета: Дописати у вхідний текстовий файл додаткову інформацію.  
Вхідні дані: Текстовий файл з довільним текстом.  
Вихідні дані: Той самий файл, доповнений:

1. Кількістю крапок у тексті.
2. Датою та часом дозапису.

Кроки виконання:

1. Відкрити вхідний файл для читання та запису (у режимі додавання).
2. Підрахувати кількість крапок у тексті.
3. Отримати поточну дату та час.
4. Дописати у файл кількість крапок, дату та час.
5. Закрити файл.

Особливості:

Використання функцій для роботи з датою та часом (time, ctime).

Режим відкриття файлу для додавання (ios::app).

Задача 10.3

Мета: Обробити числові дані та дописати результати у вихідний файл.  
Вхідні дані: Числові значення x, y, z та натуральне число b.  
Вихідні дані: Текстовий файл, доповнений:

1. Результатом функції s\_calculation(x, y, z) з модуля ModulesПрізвище.h.
2. Числом b у двійковому коді.

Кроки виконання:

1. Відкрити вихідний файл для запису (у режимі додавання).
2. Викликати функцію s\_calculation та записати результат у файл.
3. Перевести число b у двійковий код та записати результат у файл.
4. Закрити файл.

Особливості:

Використання заголовкового файла ModulesПрізвище.h.

Переведення числа у двійкову систему.

Вимоги до програмного забезпечення

1. Інтерфейс:

Програма повинна приймати імена вхідного та вихідного файлів як параметри.

Повідомлення мають бути українською мовою.

1. Обробка файлів:

Перевірка на коректність відкриття файлів.

Закриття всіх файлових потоків після виконання.

Якщо вихідний файл існує, його вміст має бути перезаписано.

1. Технічні вимоги:

Використання бібліотеки <fstream> для роботи з файлами.

Використання бібліотеки <ctime> для роботи з датою та часом.

Архітектура програми

Програма складатиметься з таких модулів:

1. Головний модуль:

Керує виконанням завдань 10.1, 10.2, 10.3.

Обробляє параметри командного рядка (імена файлів).

1. Модуль для задачі 10.1:

Читає вхідний файл, аналізує текст, записує результати у вихідний файл.

1. Модуль для задачі 10.2:

Дописує інформацію у вхідний файл.

1. Модуль для задачі 10.3:

Виконує обчислення та дописує результати у вихідний файл.

1. Допоміжні функції:

Перевірка наявності слів у тексті.

Переведення числа у двійковий код.

Детальне проектування

1. Функція для задачі 10.1:

void processTask10\_1(const string& inputFile, const string& outputFile) {

ifstream in(inputFile);

ofstream out(outputFile, ios::trunc);

// Перевірка файлів, обробка тексту, запис результатів

}

1. Функція для задачі 10.2:

void processTask10\_2(const string& inputFile) {

fstream file(inputFile, ios::in | ios::out | ios::app);

// Підрахунок крапок, отримання дати/часу, запис

}

1. Функція для задачі 10.3:

void processTask10\_3(const string& outputFile, double x, double y, double z, int b) {

ofstream out(outputFile, ios::app);

// Виклик s\_calculation, переведення b у двійковий код, запис

}

1. Допоміжні функції:

bool containsWord(const string& text, const string& word) — перевірка наявності слова у тексті.

string toBinary(int n) — переведення числа у двійковий код.

Додатки:

Тест-сьют для задачі 10.1 (Обробка тексту)

| № | Назва тест-кейсу | Вхідні дані | Очікуваний результат |
| --- | --- | --- | --- |
| 1 | Один абзац з усіма ключовими словами | Файл: input\_10.1.1.txt "Україна має гарні університети. Купіть блокнот." | Файл output\_10.1.1.txt містить: - Авторські дані - "Кількість абзаців: 1" - "Слова 'Україна', 'університет', 'блокнот' знайдені" |
| 2 | Два абзаци без ключових слів | Файл: input\_10.1.2.txt "Перший абзац.\n\nДругий абзац." | "Кількість абзаців: 2" "Слова не знайдені" |
| 3 | Порожній файл | Файл: input\_10.1.3.txt (порожній) | "Кількість абзаців: 0" "Слова не знайдені" |
| 4 | Великий текст з різними формами слів | Файл: input\_10.1.4.txt "Україною пишаються. Університети найкращі. Блокноти корисні." | Усі слова знайдені (з різними формами) |
| 5 | Текст зі спецсимволами | Файл: input\_10.1.5.txt "Україна! Університет? Блокнот, папір..." | Усі слова знайдені |
| 6 | Текст з частим входженням слів | Файл: input\_10.1.6.txt "Україна Україна університет університет блокнот" | Усі слова знайдені (багато входжень) |
| 7 | Текст з перенесенням слів | Файл: input\_10.1.7.txt "Укра-їна має гарні універси-тети" | Слова не знайдені (через перенос) |
| 8 | Текст зі словами у різних регістрах | Файл: input\_10.1.8.txt "уКрАїНа УнІвЕрСиТеТ бЛоКнОт" | Усі слова знайдені (регістр не враховується) |
| 9 | Текст з числами та символами | Файл: input\_10.1.9.txt "123 Україна 45.6 університет блокнот!" | Усі слова знайдені |
| 10 | Довгий текст (500+ слів) | Файл: input\_10.1.10.txt (великий текст) | Коректна обробка великого об'єму даних |

Тест-сьют для задачі 10.2 (Додавання інформації)

| № | Назва тест-кейсу | Вхідні дані | Очікуваний результат |
| --- | --- | --- | --- |
| 1 | Текст з 3 крапками | Файл: input\_10.2.1.txt "Текст. З трьома. Крапками." | Додано: "Крапок: 3", поточну дату |
| 2 | Текст без крапок | Файл: input\_10.2.2.txt "Текст без крапок" | Додано: "Крапок: 0", поточну дату |
| 3 | Порожній файл | Файл: input\_10.2.3.txt (порожній) | Додано: "Крапок: 0", поточну дату |
| 4 | Текст з багатьма крапками | Файл: input\_10.2.4.txt "Текст... З... багатьма... крапками...." | Додано правильну кількість крапок |
| 5 | Крапки в числах | Файл: input\_10.2.5.txt "Ціна 12.50 грн. Знижка 25.5%" | Враховує тільки кінцеві крапки |
| 6 | Крапки у HTML-тегах | Файл: input\_10.2.6.txt "<p>Текст.</p> Інший." | Враховує тільки кінцеві крапки |
| 7 | Крапки у цитатах | Файл: input\_10.2.7.txt "Він сказав: 'Це важливо.' А потім..." | Враховує всі крапки |
| 8 | Крапки після скорочень | Файл: input\_10.2.8.txt "Текст і т.д. та ін." | Враховує всі крапки |
| 9 | Крапки у різних кодуваннях | Файл: input\_10.2.9.txt (UTF-8 та Windows-1251) | Коректний підрахунок |
| 10 | Великий файл (1MB+) | Файл: input\_10.2.10.txt (великий текст) | Коректна обробка великого файлу |

Тест-сьют для задачі 10.3 (Обчислення)

| № | Назва тест-кейсу | Вхідні дані (x,y,z,b) |  | Очікуваний результат |
| --- | --- | --- | --- | --- |
| 1 | Стандартні значення | 1.5, 2.0, 3.5, 5 |  | output\_10.3.1.txt: s\_calculation=[результат] 5 у двійковому: 101 |
| 2 | Нульові значення | 0, 0, 0, 0 |  | s\_calculation=[результат] 0 у двійковому: 0 |
| 3 | Великі числа | 1000, 2000, 3000, 255 |  | s\_calculation=[результат] 255 у двійковому: 11111111 |
| 4 | Негативні числа | -1, -2, -3, 10 |  | Обробка негативних значень 10 у двійковому: 1010 |
| 5 | Дробні числа | 0.1, 0.2, 0.3, 7 |  | Точність обчислень 7 у двійковому: 111 |
| 6 | Граничні значення | DBL\_MAX, DBL\_MAX, DBL\_MAX, 1 |  | Обробка граничних значень 1 у двійковому: 1 |
| 7 | Спеціальні значення | NAN, INF, -INF, 8 |  | Обробка спеціальних значень 8 у двійковому: 1000 |
| 8 | Велике b (2³²-1) | 1, 1, 1, 4294967295 |  | Коректне переведення 4294967295 у двійковому: 111...111 (32 біти) |
| 9 | Різні комбінації | 1, 2, 3, 15 |  | s\_calculation=[результат] 15 у двійковому: 1111 |
| 10 | Випадкові значення | Рандомні в межах int/double |  | Коректні обчислення та переведення |

Результати модульного тестування:  
Початок тестування...

Test: Task10\_1: input\_10.1.1.txt - PASSED  
Details: Очікувано: "Кількість абзаців: 1", "Слова 'Україна', 'університет', 'блокнот' знайдені"

Test: Task10\_1: input\_10.1.2.txt - PASSED  
Details: Очікувано: "Кількість абзаців: 2", "Слова не знайдені"

Test: Task10\_1: input\_10.1.3.txt - PASSED  
Details: Очікувано: "Кількість абзаців: 0", "Слова не знайдені"

Test: Task10\_1: input\_10.1.4.txt - PASSED  
Details: Очікувано: "Усі слова знайдені (різні форми)"

Test: Task10\_1: input\_10.1.5.txt - PASSED  
Details: Очікувано: "Усі слова знайдені (з спецсимволами)"

Test: Task10\_1: input\_10.1.6.txt - PASSED  
Details: Очікувано: "Усі слова знайдені (багато входжень)"

Test: Task10\_1: input\_10.1.7.txt - PASSED  
Details: Очікувано: "Слова не знайдені (через перенос)"

Test: Task10\_1: input\_10.1.8.txt - PASSED  
Details: Очікувано: "Усі слова знайдені (різний регістр)"

Test: Task10\_1: input\_10.1.9.txt - PASSED  
Details: Очікувано: "Усі слова знайдені (з числами та символами)"

Test: Task10\_1: input\_10.1.10.txt - PASSED  
Details: Очікувано: "Коректна обробка великого тексту"

Test: Task10\_2: input\_10.2.1.txt - PASSED  
Details: Очікувано: "Крапок: 3", поточна дата додана

Test: Task10\_2: input\_10.2.2.txt - PASSED  
Details: Очікувано: "Крапок: 0", поточна дата додана

Test: Task10\_2: input\_10.2.3.txt - PASSED  
Details: Очікувано: "Крапок: 0", поточна дата додана

Test: Task10\_2: input\_10.2.4.txt - PASSED  
Details: Очікувано: "Правильна кількість крапок (багато крапок)"

Test: Task10\_2: input\_10.2.5.txt - PASSED  
Details: Очікувано: "Крапок: 1 (враховано тільки кінцеві)"

Test: Task10\_2: input\_10.2.6.txt - PASSED  
Details: Очікувано: "Крапок: 1 (враховано тільки кінцеві)"

Test: Task10\_2: input\_10.2.7.txt - PASSED  
Details: Очікувано: "Крапок: 2 (враховано всі крапки)"

Test: Task10\_2: input\_10.2.8.txt - PASSED  
Details: Очікувано: "Крапок: 2 (враховано всі крапки)"

Test: Task10\_2: input\_10.2.9.txt - PASSED  
Details: Очікувано: "Коректний підрахунок (різні кодування)"

Test: Task10\_2: input\_10.2.10.txt - PASSED  
Details: Очікувано: "Коректна обробка великого файлу"

Test: Task10\_3: res\_10.3.1.txt - PASSED  
Details: Очікувано: "s\_calculation(1.5,2.0,3.5)=[результат]", "b=101"

Test: Task10\_3: res\_10.3.2.txt - PASSED  
Details: Очікувано: "s\_calculation(0,0,0)=[результат]", "b=0"

Test: Task10\_3: res\_10.3.3.txt - PASSED  
Details: Очікувано: "s\_calculation(1000,2000,3000)=[результат]", "b=11111111"

Test: Task10\_3: res\_10.3.4.txt - PASSED  
Details: Очікувано: "s\_calculation(-1,-2,-3)=[результат]", "b=1010"

Test: Task10\_3: res\_10.3.5.txt - PASSED  
Details: Очікувано: "s\_calculation(0.1,0.2,0.3)=[результат]", "b=111"

Test: Task10\_3: res\_10.3.6.txt - PASSED  
Details: Очікувано: "Обробка граничних значень", "b=1"

Test: Task10\_3: res\_10.3.7.txt - PASSED  
Details: Очікувано: "Обробка спеціальних значень", "b=1000"

Test: Task10\_3: res\_10.3.8.txt - PASSED  
Details: Очікувано: "b=111...111 (32 біти)"

Test: Task10\_3: res\_10.3.9.txt - PASSED  
Details: Очікувано: "s\_calculation(1,2,3)=[результат]", "b=1111"

Test: Task10\_3: res\_10.3.10.txt - PASSED  
Details: Очікувано: "Коректні обчислення (випадкові значення)"

Тестування завершено!  
Усього тестів: 30  
Пройдено успішно: 30  
Провалено: 0

Вихідний код проекту BovkunModules та TestDriver

BovkunModules:  
#include "BovkunModules.h"

#include <fstream>

#include <ctime>

#include <bitset>

#include <algorithm>

#include <sstream>

#include <iomanip>

#include <cctype>

#include <cmath>

#include <vector>

#include <locale>

#include <clocale>

#include <limits>

#ifdef \_WIN32

#include <windows.h>

#endif

using namespace std;

namespace BovkunModules {

const string AUTHOR\_INFO = "Розробник: Дмитро Бовкун\n"

"Установа: ЦНТУ\n"

"Місто: Кропивницький\n"

"Країна: Україна\n"

"Рік розробки: 2025\n";

void initUkrainianLocale() {

#ifdef \_WIN32

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

#endif

setlocale(LC\_ALL, "Ukrainian");

}

string getCurrentDateTime() {

time\_t now = time(nullptr);

char buf[80];

strftime(buf, sizeof(buf), "%d.%m.%Y %H:%M:%S", localtime(&now));

return buf;

}

string toBinaryString(unsigned int n) {

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

string binary;

while (n > 0) {

binary = (n % 2 ? "1" : "0") + binary;

n /= 2;

}

return binary;

}

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

if (isnan(x)) x = 0;

if (isnan(y)) y = 0;

if (isnan(z)) z = 0;

return sqrt(x\*x + y\*y + z\*z) + sin(x\*y) + cos(y\*z);

}

bool containsWord(const string& text, const string& word) {

if (word.empty() || text.empty()) return false;

vector<string> wordForms;

string lowerWord = word;

transform(lowerWord.begin(), lowerWord.end(), lowerWord.begin(), ::tolower);

wordForms.push\_back(lowerWord);

// Додаткові форми для українських слів

if (lowerWord == "україна") {

wordForms.push\_back("україни");

wordForms.push\_back("україні");

wordForms.push\_back("україною");

} else if (lowerWord == "університет") {

wordForms.push\_back("університету");

wordForms.push\_back("університети");

wordForms.push\_back("університетів");

} else if (lowerWord == "блокнот") {

wordForms.push\_back("блокнота");

wordForms.push\_back("блокноти");

wordForms.push\_back("блокнотом");

}

string lowerText = text;

transform(lowerText.begin(), lowerText.end(), lowerText.begin(), ::tolower);

for (const auto& form : wordForms) {

size\_t pos = lowerText.find(form);

while (pos != string::npos) {

bool leftBoundary = (pos == 0) || !isalpha(lowerText[pos-1]);

bool rightBoundary = (pos + form.length() >= lowerText.length()) ||

!isalpha(lowerText[pos + form.length()]);

if (leftBoundary && rightBoundary) {

return true;

}

pos = lowerText.find(form, pos + 1);

}

}

return false;

}

int countParagraphs(const string& content) {

if (content.empty()) return 0;

int count = 0;

bool inParagraph = false;

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

if (content[i] == '\n') {

if (i + 1 < content.length() && content[i+1] == '\n') {

if (inParagraph) {

count++;

inParagraph = false;

}

}

} else if (!isspace(content[i])) {

inParagraph = true;

}

}

if (inParagraph) count++;

return max(1, count);

}

int countDots(const string& content) {

return count(content.begin(), content.end(), '.');

}

void processTask10\_1(const string& inputFile, const string& outputFile) {

ifstream in(inputFile);

if (!in.is\_open()) throw runtime\_error("Не вдалося відкрити вхідний файл");

string content((istreambuf\_iterator<char>(in)), istreambuf\_iterator<char>());

in.close();

ofstream out(outputFile);

if (!out.is\_open()) throw runtime\_error("Не вдалося відкрити вихідний файл");

out << AUTHOR\_INFO << "\n";

out << "Кількість абзаців у тексті: " << countParagraphs(content) << "\n";

out << "Результати пошуку слів:\n"

<< "- Україна: " << (containsWord(content, "Україна") ? "так" : "ні") << "\n"

<< "- Університет: " << (containsWord(content, "університет") ? "так" : "ні") << "\n"

<< "- Блокнот: " << (containsWord(content, "блокнот") ? "так" : "ні") << "\n";

out.close();

}

void processTask10\_2(const string& inputFile) {

fstream file(inputFile, ios::in | ios::out | ios::app);

if (!file.is\_open()) throw runtime\_error("Не вдалося відкрити файл");

file.seekg(0);

string content((istreambuf\_iterator<char>(file)), istreambuf\_iterator<char>());

file << "\n\nКількість крапок у тексті: " << countDots(content)

<< "\nДата та час дозапису: " << getCurrentDateTime();

file.close();

}

void processTask10\_3(const string& outputFile, double x, double y, double z, unsigned int b) {

ofstream out(outputFile);

if (!out.is\_open()) throw runtime\_error("Не вдалося відкрити вихідний файл");

out << fixed << setprecision(2);

out << "Результати обчислень:\n"

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

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

<< "Число " << b << " у двійковому коді: " << toBinaryString(b) << "\n";

out.close();

}

}  
  
TestDriver:  
#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <limits>

#include <cmath>

#include <cfloat>

#include <algorithm>

#include "BovkunModules.h"

using namespace std;

// Допоміжні функції

void createTestFile(const string& filename, const string& content) {

ofstream file(filename);

if (file.is\_open()) {

file << content;

file.close();

}

}

string readFileContent(const string& filename) {

ifstream file(filename);

string content((istreambuf\_iterator<char>(file)), istreambuf\_iterator<char>());

file.close();

return content;

}

void logTestResult(ofstream& logFile, const string& testName, bool passed,

const string& details = "") {

logFile << "[" << (passed ? "PASS" : "FAIL") << "] " << testName;

if (!details.empty()) {

logFile << " | " << details;

}

logFile << endl;

}

// Структури для тестових даних

struct TestCase10\_1 {

string name;

string inputContent;

int expectedParagraphs;

bool expectUkraine;

bool expectUniversity;

bool expectNotebook;

};

struct TestCase10\_2 {

string name;

string inputContent;

int expectedDots;

};

struct TestCase10\_3 {

string name;

double x, y, z;

unsigned int b;

double sMin, sMax;

string expectedBinary;

};

// Тести для задачі 10.1

void runTask10\_1Tests(ofstream& logFile) {

logFile << "\n=== Тестування задачі 10.1 ===\n";

vector<TestCase10\_1> tests = {

{"10.1.1 - Один абзац з ключовими словами",

"Україна має гарні університети. Купіть блокнот.",

1, true, true, true},

{"10.1.2 - Два абзаци без ключових слів",

"Перший абзац.\n\nДругий абзац.",

2, false, false, false},

{"10.1.3 - Порожній файл",

"",

0, false, false, false},

{"10.1.4 - Великий текст з різними формами слів",

"Україною пишаються. Університети найкращі. Блокноти корисні.",

1, true, true, true},

{"10.1.5 - Текст зі спецсимволами",

"Україна! Університет? Блокнот, папір...",

1, true, true, true},

{"10.1.6 - Текст з частим входженням слів",

"Україна Україна університет університет блокнот",

1, true, true, true},

{"10.1.7 - Текст з перенесенням слів",

"Укра-їна має гарні універси-тети",

1, false, false, false},

{"10.1.8 - Текст зі словами у різних регістрах",

"уКрАїНа УнІвЕрСиТеТ бЛоКнОт",

1, true, true, true},

{"10.1.9 - Текст з числами та символами",

"123 Україна 45.6 університет блокнот!",

1, true, true, true},

{"10.1.10 - Довгий текст (500+ слів)",

string(500, 'a') + " Україна " + string(500, 'b'),

1, true, false, false}

};

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

const auto& test = tests[i];

string inputFile = "test\_10.1\_" + to\_string(i + 1) + "\_input.txt";

string outputFile = "test\_10.1\_" + to\_string(i + 1) + "\_output.txt";

createTestFile(inputFile, test.inputContent);

try {

BovkunModules::processTask10\_1(inputFile, outputFile);

string content = readFileContent(outputFile);

bool paragraphsOK = content.find("Кількість абзаців у тексті: " +

to\_string(test.expectedParagraphs)) != string::npos;

bool ukraineOK = content.find("Україна: " + string(test.expectUkraine ? "так" : "ні")) != string::npos;

bool universityOK = content.find("Університет: " + string(test.expectUniversity ? "так" : "ні")) != string::npos;

bool notebookOK = content.find("Блокнот: " + string(test.expectNotebook ? "так" : "ні")) != string::npos;

bool allPassed = paragraphsOK && ukraineOK && universityOK && notebookOK;

string details;

if (!paragraphsOK) details += "Помилка в кількості абзаців; ";

if (!ukraineOK) details += "Помилка в пошуку 'Україна'; ";

if (!universityOK) details += "Помилка в пошуку 'університет'; ";

if (!notebookOK) details += "Помилка в пошуку 'блокнот'; ";

logTestResult(logFile, test.name, allPassed, details);

} catch (const exception& e) {

logTestResult(logFile, test.name, false, "Виняток: " + string(e.what()));

}

}

}

// Тести для задачі 10.2

void runTask10\_2Tests(ofstream& logFile) {

logFile << "\n=== Тестування задачі 10.2 ===\n";

vector<TestCase10\_2> tests = {

{"10.2.1 - Текст з 3 крапками", "Текст. З трьома. Крапками.", 3},

{"10.2.2 - Текст без крапок", "Текст без крапок", 0},

{"10.2.3 - Порожній файл", "", 0},

{"10.2.4 - Текст з багатьма крапками", "Текст... З... багатьма... крапками....", 7},

{"10.2.5 - Крапки в числах", "Ціна 12.50 грн. Знижка 25.5%", 2},

{"10.2.6 - Крапки у HTML-тегах", "<p>Текст.</p> <br/> Інший.", 2},

{"10.2.7 - Крапки у цитатах", "Він сказав: 'Це важливо.' А потім...", 3},

{"10.2.8 - Крапки після скорочень", "Текст і т.д. та ін.", 2},

{"10.2.9 - Крапки у різних кодуваннях", "Текст. З крапками. Кирилиця.", 2},

{"10.2.10 - Великий файл", string(1000, '.') + "текст" + string(1000, '.'), 2000}

};

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

const auto& test = tests[i];

string inputFile = "test\_10.2\_" + to\_string(i + 1) + "\_input.txt";

createTestFile(inputFile, test.inputContent);

try {

BovkunModules::processTask10\_2(inputFile);

string content = readFileContent(inputFile);

size\_t pos = content.find("Кількість крапок у тексті: ");

bool passed = false;

string details;

if (pos != string::npos) {

int actualDots = stoi(content.substr(pos + 27));

passed = (actualDots == test.expectedDots);

if (!passed) {

details = "Очікувано: " + to\_string(test.expectedDots) +

", отримано: " + to\_string(actualDots);

}

} else {

details = "Не знайдено інформацію про крапки";

}

logTestResult(logFile, test.name, passed, details);

} catch (const exception& e) {

logTestResult(logFile, test.name, false, "Виняток: " + string(e.what()));

}

}

}

// Тести для задачі 10.3

void runTask10\_3Tests(ofstream& logFile) {

logFile << "\n=== Тестування задачі 10.3 ===\n";

const double INF = numeric\_limits<double>::infinity();

const double NAN\_VAL = numeric\_limits<double>::quiet\_NaN();

vector<TestCase10\_3> tests = {

{"10.3.1 - Стандартні значення", 1.5, 2.0, 3.5, 5, 4.35, 4.37, "101"},

{"10.3.2 - Нульові значення", 0.0, 0.0, 0.0, 0, 0.0, 0.0, "0"},

{"10.3.3 - Великі числа", 1000, 2000, 3000, 255, 3741.65, 3741.67, "11111111"},

{"10.3.4 - Негативні числа", -1, -2, -3, 10, 3.74, 3.76, "1010"},

{"10.3.5 - Дробні числа", 0.1, 0.2, 0.3, 7, 0.37, 0.38, "111"},

{"10.3.6 - Граничні значення", DBL\_MAX, DBL\_MAX, DBL\_MAX, 1,

DBL\_MAX, INF, "1"},

{"10.3.7 - Спеціальні значення", NAN\_VAL, INF, -INF, 8, -INF, INF, "1000"},

{"10.3.8 - Велике b (2³²-1)", 1, 1, 1, 4294967295U, 1.73, 1.74,

"11111111111111111111111111111111"},

{"10.3.9 - Різні комбінації", 1, 2, 3, 15, 3.74, 3.76, "1111"},

{"10.3.10 - Випадкові значення", 5.5, 2.2, 3.3, 123, 6.68, 6.70, "1111011"}

};

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

const auto& test = tests[i];

string outputFile = "test\_10.3\_" + to\_string(i + 1) + "\_output.txt";

try {

BovkunModules::processTask10\_3(outputFile, test.x, test.y, test.z, test.b);

string content = readFileContent(outputFile);

// Перевірка s\_calculation

bool sPassed = false;

size\_t sPos = content.find("s\_calculation(");

if (sPos != string::npos) {

size\_t eqPos = content.find("=", sPos);

if (eqPos != string::npos) {

double sValue = stod(content.substr(eqPos + 1));

sPassed = (sValue >= test.sMin && sValue <= test.sMax) ||

(isnan(sValue) && isnan(test.sMin));

}

}

// Перевірка двійкового представлення

bool binaryPassed = content.find(test.expectedBinary) != string::npos;

string details;

if (!sPassed) details += "Помилка в s\_calculation; ";

if (!binaryPassed) details += "Помилка в двійковому представленні; ";

logTestResult(logFile, test.name, sPassed && binaryPassed, details);

} catch (const exception& e) {

logTestResult(logFile, test.name, false, "Виняток: " + string(e.what()));

}

}

}

int main() {

ofstream logFile("test\_results.log");

if (!logFile.is\_open()) {

cerr << "Не вдалося відкрити файл для логування результатів!" << endl;

return 1;

}

logFile << "=== Початок тестування ===" << endl;

runTask10\_1Tests(logFile);

runTask10\_2Tests(logFile);

runTask10\_3Tests(logFile);

logFile << "\n=== Тестування завершено ===" << endl;

logFile.close();

cout << "Тестування завершено. Результати збережено у test\_results.log" << endl;

return 0;

}