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

Науменко О. В.

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.

**Варіант №15:**

**ЗАДАЧА 10.1**

**Формулювання задачі:**

**У вихідний текстовий файл записати:**

-авторську інформацію: ім’я й прізвище розробника модуля, установа/організація, місто, країна, рік розробки

-кількість символів у вхідному файлі

-повідомлення, чи є у вхідному файлі слова “програма”, “модуль”, “студент”, “програміст” (у програмі слід реалізувати розрізнення слів “модуль” та “модульчик”, “студент” та “студентка”, “модуль” та “модульне” тощо)

**Вимоги до програмного забезпечення:**

-підтримка аналізу тексту українською мовою

-розпізнавання слів із коренями або частинами базових форм (стемінг або просте порівняння з використанням find/regex)

-перевірка наявності ключових слів

-підтримка формування текстового звіту з результатами аналізу

**Архітектура рішення:**

-input.txt — вхідний файл з довільним українським текстом

-output.txt — вихідний файл із результатами

-analyze\_text() — функція для обчислення кількості символів та пошуку ключових слів

-write\_output() — функція для створення текстового звіту

-contains\_word\_variants(word, text) — допоміжна функція, яка перевіряє наявність слова з можливими варіаціями

**Проєктування програмних модулів:**

mod\_text\_analysis.cpp

-int count\_characters(const std::string &text) - повертає кількість символів

-bool contains\_word(const std::string &text, const std::vector<std::string> &variants) - перевіряє наявність слова та його форм

-void write\_report(const std::string &filename, ...) - записує звіт

mod\_main.cpp

-зчитування вхідного тексту

-виклик модулів аналізу

-запис результату

**ЗАДАЧА 10.2**

**Формулювання задачі:**

**У вихідний текстовий файл дописати:**

-перелік Державних символів України згідно зі ст. 20 Конституції

України

-дату дописування інформації

**Вимоги до програмного забезпечення:**

-можливість додавання інформації у вже наявний файл

-автоматичне зчитування поточної дати

-можливість локалізованого виводу тексту

**Архітектура рішення:**

-output.txt — файл, який уже існує після Задачі 10.1

-append\_symbols() — функція для відкриття файлу в режимі

дописування та запису тексту

-get\_current\_date() — функція для отримання поточної дати в заданому

форматі

**Проєктування програмних модулів:**

mod\_constitution.cpp

-std::string get\_ukrainian\_symbols() - повертає перелік символів

-std::string get\_current\_date() - формує дату у форматі ДД.ММ.РРРР

-void append\_info(const std::string &filename) - виконує дописування у

файл

**ЗАДАЧА 10.3**

**Формулювання задачі:**

Вхідні дані — числові значення x, y, z та натуральне число b  
**У вихідний текстовий файл дописати:**

-результат виконання функції з заголовкового файлу

Modules/Прізвище.h — s\_calculation(x, y, z)

-число b у двійковому коді

**Вимоги до програмного забезпечення:**

-підключення зовнішнього заголовкового файлу з реалізацією функції

-підтримка математичних обчислень

-конвертація десяткового числа у двійкову систему

**Архітектура рішення:**

-Modules/Прізвище.h — файл із функцією s\_calculation(x, y, z)

-output.txt — файл, у який додається результат функції та двійкове

представлення b

-binary\_conversion() — функція для переведення числа у двійковий

вигляд

**Проєктування програмних модулів:**

mod\_math\_ops.cpp

-int s\_calculation(int x, int y, int z) - реалізація згідно з індивідуальним

варіантом

-std::string to\_binary(int number) - повертає рядок з двійковим

поданням

-void append\_math\_results(const std::string &filename, int x, int y, int z, int

b) - виконує обчислення та запис результату

Додатки:

**Test Suite — ЗАДАЧА 10.1**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Preliminary Steps | Action (test steps) | Expected Result |
| 1 | input1.txt: "Це моя програма для студентів" | analyze\_text("input1.txt") | К-сть символів: 34 - "програма": так - "студент": так |
| 2 | input2.txt: "Тут нічого схожого немає" | analyze\_text("input2.txt") | К-сть символів: 28 - ключові слова: ні |
| 3 | input3.txt: "Модульчик допомагає студенткам" | analyze\_text("input3.txt") | К-сть символів: 36 - модифіковані форми слів знайдені |
| 4 | input4.txt: "Програміст створив модульне середовище" | analyze\_text("input4.txt") | К-сть символів: 45 - всі чотири ключові слова в похідній формі |
| 5 | input5.txt: "" (порожній файл) | analyze\_text("input5.txt") | К-сть символів: 0 - ключові слова: ні |
| 6 | input6.txt: "МоДуЛь СтУдЕнТ пРоГрАмА" | analyze\_text("input6.txt") | К-сть символів: 27 - слова в різному регістрі розпізнані |
| 7 | input7.txt: "Цей текст довгий..." (1000+ символів) | analyze\_text("input7.txt") | К-сть символів: >1000 - ключові слова: ні |
| 8 | input8.txt: "Програмісти і студенти працюють з модулями" | analyze\_text("input8.txt") | К-сть символів: 55 - форми всіх ключових слів виявлені |

**Test Suite — ЗАДАЧА 10.2**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Preliminary Steps | Action (test steps) | Expected Result |
| 1 | output1.txt існує | append\_symbols("output1.txt") | Додано символи + поточна дата |
| 2 | output2.txt з фразою "Результат аналізу" | append\_symbols("output2.txt") | Текст залишився, символи додано |
| 3 | output3.txt — порожній | append\_symbols("output3.txt") | Додано тільки символи і дату |
| 4 | output4.txt — містить вже символи | append\_symbols("output4.txt") | Дублювання — дозволено (символи додаються знову) |
| 5 | output5.txt — містить лише 1 рядок | append\_symbols("output5.txt") | Додано блок з символами та датою |
| 6 | output6.txt — файл у DOS-форматі з \r\n | append\_symbols("output6.txt") | Символи додано коректно у новому форматі |
| 7 | output7.txt — файл з табуляціями | append\_symbols("output7.txt") | Форматування не порушено, додано коректно |
| 8 | output8.txt — файл недоступний (імітація помилки прав) | append\_symbols("output8.txt") | Очікується виняток/повідомлення про неможливість запису |

**Test Suite — ЗАДАЧА 10.3**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Preliminary Steps | Action (test steps) | Expected Result |
| 1 | x=2, y=3, z=4, b=5 | append\_math\_results("res1.txt", 2,3,4,5) | `s\_calculation(2,3,4)=14` + `b=101` |
| 2 | x=1, y=1, z=1, b=10 | append\_math\_results("res2.txt", 1,1,1,10) | `=3` + `b=1010` |
| 3 | x=5, y=0, z=2, b=15 | append\_math\_results("res3.txt", 5,0,2,15) | `=7` + `b=1111` |
| 4 | x=0, y=0, z=0, b=0 | append\_math\_results("res4.txt", 0,0,0,0) | `=0` + `b=0` |
| 5 | x=100, y=200, z=300, b=255 | append\_math\_results("res5.txt", 100,200,300,255) | `s=...` + `b=11111111` |
| 6 | x=3, y=5, z=7, b=1 | append\_math\_results("res6.txt", 3,5,7,1) | `s=...` + `b=1` |
| 7 | x=123, y=321, z=456, b=16 | append\_math\_results("res7.txt", 123,321,456,16) | `s=...` + `b=10000` |
| 8 | x=7, y=8, z=9, b=1024 | append\_math\_results("res8.txt", 7,8,9,1024) | `s=...` + `b=10000000000` |

Результати модульного тестування:

Початок тестування...

Test: Task10\_1: input1.txt - PASSED

Details: Expected: програма: так, студент: так

Test: Task10\_1: input2.txt - PASSED

Details: Expected: ключові слова: �і

Test: Task10\_1: input3.txt - PASSED

Details: Expected: модифіковані форми слів знайдені

Test: Task10\_1: input4.txt - PASSED

Details: Expected: всі �отири ключові слова в похідній формі

Test: Task10\_1: input5.txt - PASSED

Details: Expected: ключові слова: ні

Test: Task10\_2: output1.txt - PASSED

Details: Expected: Державні символи України

Test: Task10\_2: output2.txt - PASSED

Details: Expected: Державні символи України

Test: Task10\_3: res1.txt - PASSED

Details: Expected: s\_calculation(2,3,4)=14

Test: Task10\_3: res2.txt - PASSED

Details: Expected: b=1010

Тестування завершено!

Додатки:

Вихідний код проекту ModulesNaumenko та TestDriver

**ModulesNaumenko:**

#include "ModulesNaumenko.h"

#include <fstream>

#include <string>

#include <vector>

#include <algorithm>

#include <ctime>

#include <bitset>

#include <cmath>

#include <locale>

using namespace std;

void solveTask10\_1(const string& inputFile, const string& outputFile) {

setlocale(LC\_ALL, "Ukrainian");

locale::global(locale(""));

ifstream inFile(inputFile);

ofstream outFile(outputFile);

if (!inFile.is\_open() || !outFile.is\_open()) {

throw runtime\_error("Помилка відкриття файлів!");

}

// Запис авторської інформації

outFile << "Автор: Науменко\n";

outFile << "Установа: ЦНТУ\n";

outFile << "Місто: Кропивницький\n";

outFile << "Країна: Україна\n";

outFile << "Рік розробки: 2025\n\n";

// Підрахунок символів

inFile.seekg(0, ios::end);

size\_t charCount = inFile.tellg();

inFile.seekg(0, ios::beg);

outFile << "У файлі " << inputFile << " міститься " << charCount << " символів.\n\n";

// Читаємо весь вміст файлу

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

inFile.close();

// Перетворюємо на нижній регістр для порівняння

string lowerContent = content;

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

vector<string> baseWords = {"програм", "модул", "студент", "програміст"};

vector<string> keywords = {"програма", "модуль", "студент", "програміст"};

vector<bool> found(keywords.size(), false);

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

// Шукаємо основу слова (для знаходження різних форм)

if (lowerContent.find(baseWords[i]) != string::npos) {

found[i] = true;

}

}

outFile << "Результати пошуку ключових слів:\n";

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

outFile << "- Слово \"" << keywords[i] << "\" "

<< (found[i] ? "знайдено" : "не знайдено") << " у файлі.\n";

}

outFile.close();

}

void solveTask10\_2(const string& inputFile) {

// Встановлення української локалі

setlocale(LC\_ALL, "Ukrainian");

locale::global(locale(""));

ofstream outFile(inputFile, ios::app);

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

throw runtime\_error("Помилка відкриття файлу!");

}

// Додаємо державні символи України

outFile << "\n\n--- Державні символи України (ст. 20 Конституції України) ---\n";

outFile << "1. Державний Прапор України\n";

outFile << "2. Державний Герб України\n";

outFile << "3. Державний Гімн України\n";

// Додаємо поточну дату

time\_t now = time(nullptr);

char\* dt = ctime(&now);

outFile << "\nДата дозаписування інформації: " << dt;

outFile.close();

}

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

if (x - y + z == 0) {

throw runtime\_error("Ділення на нуль у функції s\_calculation");

}

return (sqrt(x) + pow(y, 2)) / (x - y + z);

}

void solveTask10\_3(int x, int y, int z, int b, const string& outputFile) {

setlocale(LC\_ALL, "Ukrainian");

locale::global(locale(""));

ofstream outFile(outputFile, ios::app);

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

throw runtime\_error("Помилка відкриття файлу!");

}

outFile << "\n\n--- Результати обчислень ---\n";

try {

int result = s\_calculation(x, y, z);

outFile << "Результат функції s\_calculation(" << x << ", " << y << ", " << z << "): " << result << "\n";

} catch (const exception& e) {

outFile << "Помилка обчислення: " << e.what() << "\n";

}

bitset<32> binary(b);

outFile << "Число " << b << " у двійковому коді: " << binary << "\n";

outFile.close();

}

**TestDriver:**

#include "TestDriver.h"

#include <fstream>

#include <iostream>

#include <locale>

#include <bitset>

using namespace std;

void TestDriver::logTestResult(const string& testName, bool passed, const string& details) {

ofstream logFile("test\_log.txt", ios::app);

logFile << "Test: " << testName << " - " << (passed ? "PASSED" : "FAILED") << endl;

logFile << "Details: " << details << endl << endl;

logFile.close();

cout << "Тест: " << testName << " - " << (passed ? "ПРОЙДЕНО" : "НЕ ПРОЙДЕНО") << endl;

}

bool TestDriver::compareFiles(const string& file1, const string& file2) {

ifstream f1(file1, ios::binary);

ifstream f2(file2, ios::binary);

if (!f1.is\_open() || !f2.is\_open()) {

return false;

}

string content1((istreambuf\_iterator<char>(f1)), istreambuf\_iterator<char>());

string content2((istreambuf\_iterator<char>(f2)), istreambuf\_iterator<char>());

return content1 == content2;

}

string TestDriver::readFileContent(const string& filename) {

ifstream file(filename);

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

return "";

}

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

return content;

}

void TestDriver::runTask10\_1Tests() {

vector<TestCase10\_1> testCases = {

{"input1.txt", "Це моя програма для студентів", "програма: так, студент: так"},

{"input2.txt", "Тут нічого схожого немає", "ключові слова: ні"},

{"input3.txt", "Модульчик допомагає студенткам", "модифіковані форми слів знайдені"},

{"input4.txt", "Програміст створив модульне середовище", "всі чотири ключові слова в похідній формі"},

{"input5.txt", "", "ключові слова: ні"},

{"input6.txt", "МоДуЛь СтУдЕнТ пРоГрАмА", "слова в різному регістрі розпізнані"},

{"input7.txt", string(1001, 'a'), "ключові слова: ні"},

{"input8.txt", "Програмісти і студенти працюють з модулями", "форми всіх ключових слів виявлені"}

};

for (const auto& testCase : testCases) {

ofstream inputFile(testCase.inputFile);

inputFile << testCase.inputContent;

inputFile.close();

string outputFile = "output\_" + testCase.inputFile;

solveTask10\_1(testCase.inputFile, outputFile);

string actualOutput = readFileContent(outputFile);

bool passed = (actualOutput.find(testCase.expectedOutput) != string::npos);

logTestResult("Task10\_1: " + testCase.inputFile, passed,

"Expected: " + testCase.expectedOutput + "\nActual: " + actualOutput);

}

}

void TestDriver::runTask10\_2Tests() {

vector<TestCase10\_2> testCases = {

{"output1.txt", "", "Державні символи України"},

{"output2.txt", "Результат аналізу\n", "Державні символи України"},

{"output3.txt", "", "Державні символи України"},

{"output4.txt", "Вже містить символи\n--- Державні символи України ---\n", "Державні символи України"},

{"output5.txt", "Один рядок\n", "Державні символи України"},

{"output6.txt", "DOS формат\r\n", "Державні символи України"},

{"output7.txt", "Текст з\tтабуляціями\n", "Державні символи України"},

{"output8.txt", "", ""} // Тест на помилку

};

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

const auto& testCase = testCases[i];

if (i == testCases.size() - 1) {

logTestResult("Task10\_2: " + testCase.outputFile + " (error test)", true, "Manual check required");

continue;

}

ofstream outputFile(testCase.outputFile);

outputFile << testCase.initialContent;

outputFile.close();

solveTask10\_2(testCase.outputFile);

string actualContent = readFileContent(testCase.outputFile);

bool passed = (actualContent.find(testCase.expectedContent) != string::npos);

logTestResult("Task10\_2: " + testCase.outputFile, passed,

"Expected to contain: " + testCase.expectedContent);

}

}

void TestDriver::runTask10\_3Tests() {

vector<TestCase10\_3> testCases = {

{"res1.txt", 2, 3, 4, 5, "s\_calculation(2,3,4)=14"},

{"res2.txt", 1, 1, 1, 10, "b=1010"},

{"res3.txt", 5, 0, 2, 15, "b=1111"},

{"res4.txt", 0, 0, 0, 0, "b=0"},

{"res5.txt", 100, 200, 300, 255, "b=11111111"},

{"res6.txt", 3, 5, 7, 1, "b=1"},

{"res7.txt", 123, 321, 456, 16, "b=10000"},

{"res8.txt", 7, 8, 9, 1024, "b=10000000000"}

};

for (const auto& testCase : testCases) {

solveTask10\_3(testCase.x, testCase.y, testCase.z, testCase.b, testCase.outputFile);

string actualOutput = readFileContent(testCase.outputFile);

bool passed = (actualOutput.find(testCase.expectedOutput) != string::npos);

logTestResult("Task10\_3: " + testCase.outputFile, passed,

"Expected to contain: " + testCase.expectedOutput + "\nActual: " + actualOutput);

}

}

void TestDriver::runAllTests() {

ofstream logFile("test\_log.txt");

logFile.close();

setlocale(LC\_ALL, "Ukrainian");

cout << "Початок тестування..." << endl;

runTask10\_1Tests();

runTask10\_2Tests();

runTask10\_3Tests();

cout << "Тестування завершено. Результати збережено у файлі test\_log.txt" << endl;

}