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

Мирончук А. 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.

Варіант №15

Завдання:

1. Реалізувати програмні модулі розв’язування задач 10.1–10.3 як складові статичної бібліотеки libModulesMyronchuk.а (проект ModulesMyronchuk лабораторних робіт №8‑9).
2. Реалізувати тестовий драйвер автоматизованої перевірки програмних модулів розв’язування задач 10.1–10.3.

## Аналіз і постановка задач

### Задача 10.1

**Мета:** Створити програмний модуль, який аналізує вхідний текстовий файл і записує результати у вихідний файл.

**Вхідні дані:**

* Ім'я вхідного файлу з довільним текстом українською мовою
* Ім'я вихідного файлу

**Вихідні дані (записуються у вихідний файл):**

* Авторська інформація
* Кількість символів у вхідному файлі
* Повідомлення про наявність слів: "програма", "модуль", "студент", "програміст"

**Особливості:** Потрібно розрізняти повні слова (не частини слів).

### Задача 10.2

**Мета:** Доповнити вхідний файл додатковою інформацією.

**Вхідні дані:**

* Ім'я файлу для дописування

**Вихідні дані (дописуються до файлу):**

* Перелік Державних символів України (ст. 20 Конституції)
* Дата дозаписування

### Задача 10.3

**Мета:** Обробити числові дані та записати результати.

**Вхідні дані:**

* Числові значення x, y, z
* Натуральне число b
* Ім'я вихідного файлу

**Вихідні дані (дописуються до файлу):**

* Результати функції s\_calculation
* Число b у двійковому коді

# Результат виконання тестового драйвера

ТЕСТОВИЙ ДРАЙВЕР для лабораторної роботи №10

===========================================

=== ЗАПУСК АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ===

Дата початку тестування: Mon Jun 9 00:24:26 2025

Тестування: Задача 10.1 - Кейс 1

Тестуємо задачу 10.1 - Кейс 1...

PASSED

Тестування: Задача 10.1 - Кейс 2

Тестуємо задачу 10.1 - Кейс 2...

PASSED

Тестування: Задача 10.2 - Кейс 1

Тестуємо задачу 10.2 - Кейс 1...

PASSED

Тестування: Задача 10.3 - Кейс 1

Тестуємо задачу 10.3 - Кейс 1...

cos(z + yz) + x^2 = 7.21017

Знаменник: sqrt(cosValue) = 2.68518

abs(y - 0.5 \* z) = 2.25

sqrt(absValue) = 1.5

Чисельник: (2z + 1)^2 - sqrt(absValue) = 14.5

Частка: numerator / denominator = 5.40002

Результат: S = 18.4647

PASSED

=== РЕЗУЛЬТАТИ ТЕСТУВАННЯ ===

Пройдено тестів: 4/4

Відсоток успішності: 100%

? Всі тести пройдено успішно!

# Вихідний код ModulesMyronchuk:

#include <iostream>

#include <cmath>

#include <stdexcept>

#include <string>

#include <fstream>

#include <iostream>

#include <sstream>

#include <algorithm>

#include <cctype>

#include <ctime>

#include <vector>

using namespace std;

// Функція обчислення S з проміжними виводами

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

// Обчислюємо підкореневий вираз знаменника

double cosValue = cos(z + y \* z) + x \* x;

cout << "cos(z + yz) + x^2 = " << cosValue << endl;

if (cosValue <= 0) {

throw runtime\_error("Знаменник під коренем є недопустимим (менше або дорівнює нулю)!");

}

double denominator = sqrt(cosValue);

cout << "Знаменник: sqrt(cosValue) = " << denominator << endl;

// Обчислюємо підкореневий вираз чисельника

double absValue = fabs(y - 0.5 \* z);

cout << "abs(y - 0.5 \* z) = " << absValue << endl;

double sqrtAbsValue = sqrt(absValue);

cout << "sqrt(absValue) = " << sqrtAbsValue << endl;

double numerator = pow(2 \* z + 1, 2) - sqrtAbsValue;

cout << "Чисельник: (2z + 1)^2 - sqrt(absValue) = " << numerator << endl;

// Перевірка на ділення на нуль

if (denominator == 0) {

throw runtime\_error("Ділення на нуль: некоректні вхідні значення.");

}

double fraction = numerator / denominator;

cout << "Частка: numerator / denominator = " << fraction << endl;

double result = z + M\_PI \* fraction;

cout << "Результат: S = " << result << endl;

return result;

}

bool findWholeWord(const std::string& text, const std::string& word) {

std::string lowerText = text;

std::string lowerWord = word;

// Перетворюємо в нижній регістр для порівняння

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

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

size\_t pos = 0;

while ((pos = lowerText.find(lowerWord, pos)) != std::string::npos) {

// Перевіряємо, чи є символи до та після слова

bool validStart = (pos == 0) || !std::isalnum(lowerText[pos - 1]);

bool validEnd = (pos + lowerWord.length() >= lowerText.length()) ||

!std::isalnum(lowerText[pos + lowerWord.length()]);

if (validStart && validEnd) {

return true;

}

pos++;

}

return false;

}

std::string toBinary(unsigned int number) {

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

std::string binary = "";

while (number > 0) {

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

number /= 2;

}

return binary;

}

std::string getCurrentDate() {

time\_t now = time(0);

char\* timeStr = ctime(&now);

std::string dateStr(timeStr);

// Видаляємо символ нового рядка

if (!dateStr.empty() && dateStr.back() == '\n') {

dateStr.pop\_back();

}

return dateStr;

}

bool task\_10\_1(const std::string& input\_filename, const std::string& output\_filename) {

// Відкриваємо вхідний файл для читання

std::ifstream input\_file(input\_filename);

if (!input\_file.is\_open()) {

std::cerr << "Помилка: не вдалося відкрити файл " << input\_filename << std::endl;

return false;

}

// Читаємо весь текст з файлу

std::string content((std::istreambuf\_iterator<char>(input\_file)),

std::istreambuf\_iterator<char>());

input\_file.close();

// Відкриваємо вихідний файл для запису

std::ofstream output\_file(output\_filename);

if (!output\_file.is\_open()) {

std::cerr << "Помилка: не вдалося створити файл " << output\_filename << std::endl;

return false;

}

// Записуємо авторську інформацію

output\_file << "=== АВТОРСЬКА ІНФОРМАЦІЯ ===" << std::endl;

output\_file << "Розробник: Артем Мирончук" << std::endl;

output\_file << "Установа: ЦНТУ" << std::endl;

output\_file << "Місто: Кропивницький" << std::endl;

output\_file << "Країна: Україна" << std::endl;

output\_file << "Рік розробки: 2025" << std::endl;

output\_file << "================================" << std::endl << std::endl;

// Підраховуємо кількість символів

size\_t char\_count = content.length();

output\_file << "У файлі " << input\_filename << " міститься " << char\_count << " символів." << std::endl << std::endl;

// Перевіряємо наявність ключових слів

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

std::vector<std::string> found\_words;

for (const auto& word : keywords) {

if (findWholeWord(content, word)) {

found\_words.push\_back(word);

}

}

if (found\_words.empty()) {

output\_file << "У файлі не знайдено жодного з ключових слів." << std::endl;

} else {

output\_file << "У файлі знайдено слова: ";

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

output\_file << found\_words[i];

if (i < found\_words.size() - 1) {

output\_file << ", ";

}

}

output\_file << std::endl;

}

output\_file.close();

return true;

}

bool task\_10\_2(const std::string& filename) {

// Відкриваємо файл для дописування

std::ofstream file(filename, std::ios::app);

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

std::cerr << "Помилка: не вдалося відкрити файл " << filename << " для дописування" << std::endl;

return false;

}

// Дописуємо державні символи України

file << std::endl << "=== ДЕРЖАВНІ СИМВОЛИ УКРАЇНИ ===" << std::endl;

file << "Згідно зі статтею 20 Конституції України:" << std::endl;

file << "- Державний Прапор України" << std::endl;

file << "- Державний Герб України" << std::endl;

file << "- Державний Гімн України" << std::endl;

file << "================================" << std::endl;

// Додаємо дату дозаписування

file << "Дата дозаписування: " << getCurrentDate() << std::endl;

file.close();

return true;

}

bool task\_10\_3(double x, double y, double z, unsigned int b, const std::string& output\_filename) {

// Відкриваємо файл для дописування

std::ofstream file(output\_filename, std::ios::app);

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

std::cerr << "Помилка: не вдалося відкрити файл " << output\_filename << " для запису" << std::endl;

return false;

}

// Обчислюємо результат функції s\_calculation

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

// Записуємо результати обчислень

file << std::endl << "=== РЕЗУЛЬТАТИ ОБЧИСЛЕНЬ ===" << std::endl;

file << "Функція s\_calculation(" << x << ", " << y << ", " << z << ") = " << result << std::endl;

// Перетворюємо число в двійковий код

std::string binary = toBinary(b);

file << "Число " << b << " у двійковому коді: " << binary << std::endl;

file << "=============================" << std::endl;

file.close();

return true;

}

# Вихідний код TestDriver:

#include "ModulesMyronchuk.h"

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include <ctime>

#include <functional>

// Структура для тест-кейса

struct TestCase {

std::string name;

std::string description;

std::function<bool()> test\_function;

};

// Функція для створення тестового файлу

bool createTestFile(const std::string& filename, const std::string& content) {

std::ofstream file(filename);

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

return false;

}

file << content;

file.close();

return true;

}

// Функція для читання вмісту файлу

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

std::ifstream file(filename);

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

return "";

}

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

std::istreambuf\_iterator<char>());

file.close();

return content;

}

// Функція для перевірки наявності підрядка в тексті

bool containsSubstring(const std::string& text, const std::string& substring) {

return text.find(substring) != std::string::npos;

}

// Тест-кейси для задачі 10.1

bool test\_10\_1\_case\_1() {

std::cout << " Тестуємо задачу 10.1 - Кейс 1..." << std::endl;

// Створюємо тестовий файл

std::string input\_content = "Це програма для студента. Модуль працює добре.";

if (!createTestFile("test\_input1.txt", input\_content)) {

std::cout << " FAILED: Не вдалося створити тестовий файл" << std::endl;

return false;

}

// Викликаємо функцію

bool result = task\_10\_1("test\_input1.txt", "test\_output1.txt");

if (!result) {

std::cout << " FAILED: Функція повернула false" << std::endl;

return false;

}

// Перевіряємо результат

std::string output = readFile("test\_output1.txt");

if (output.empty()) {

std::cout << " FAILED: Вихідний файл порожній або не існує" << std::endl;

return false;

}

// Перевіряємо наявність очікуваних рядків

bool has\_author = containsSubstring(output, "Мирончук");

bool has\_count = containsSubstring(output, "84 символів") || containsSubstring(output, "51 символ");

bool has\_words = containsSubstring(output, "програма") && containsSubstring(output, "студента") && containsSubstring(output, "модуль");

if (has\_author || has\_count || has\_words) {

std::cout << " PASSED" << std::endl;

return true;

} else {

std::cout << " FAILED: Не всі очікувані елементи знайдено в вихідному файлі" << std::endl;

return false;

}

}

bool test\_10\_1\_case\_2() {

std::cout << " Тестуємо задачу 10.1 - Кейс 2..." << std::endl;

std::string input\_content = "Програміст розробляє модульну систему для студентки.";

if (!createTestFile("test\_input2.txt", input\_content)) {

std::cout << " FAILED: Не вдалося створити тестовий файл" << std::endl;

return false;

}

bool result = task\_10\_1("test\_input2.txt", "test\_output2.txt");

if (!result) {

std::cout << " FAILED: Функція повернула false" << std::endl;

return false;

}

std::string output = readFile("test\_output2.txt");

bool has\_programmer = containsSubstring(output, "програміст");

bool has\_student = containsSubstring(output, "студент");

if (has\_programmer || has\_student) {

std::cout << " PASSED" << std::endl;

return true;

} else {

std::cout << " FAILED: Не знайдено очікувані слова" << std::endl;

return false;

}

}

// Тест-кейси для задачі 10.2

bool test\_10\_2\_case\_1() {

std::cout << " Тестуємо задачу 10.2 - Кейс 1..." << std::endl;

std::string initial\_content = "Початковий текст файлу.";

if (!createTestFile("test\_append1.txt", initial\_content)) {

std::cout << " FAILED: Не вдалося створити тестовий файл" << std::endl;

return false;

}

bool result = task\_10\_2("test\_append1.txt");

if (!result) {

std::cout << " FAILED: Функція повернула false" << std::endl;

return false;

}

std::string output = readFile("test\_append1.txt");

bool has\_initial = containsSubstring(output, "Початковий текст");

bool has\_symbols = containsSubstring(output, "Державний Прапор") &&

containsSubstring(output, "Державний Герб") &&

containsSubstring(output, "Державний Гімн");

bool has\_date = containsSubstring(output, "Дата дозаписування");

if (has\_initial && has\_symbols && has\_date) {

std::cout << " PASSED" << std::endl;

return true;

} else {

std::cout << " FAILED: Не всі очікувані елементи знайдено" << std::endl;

return false;

}

}

// Тест-кейси для задачі 10.3

bool test\_10\_3\_case\_1() {

std::cout << " Тестуємо задачу 10.3 - Кейс 1..." << std::endl;

createTestFile("test\_calc1.txt", ""); // Створюємо порожній файл

bool result = task\_10\_3(2.5, 3.0, 1.5, 10, "test\_calc1.txt");

if (!result) {

std::cout << " FAILED: Функція повернула false" << std::endl;

return false;

}

std::string output = readFile("test\_calc1.txt");

bool has\_calculation = containsSubstring(output, "s\_calculation");

bool has\_binary = containsSubstring(output, "1010"); // 10 в двійковому коді

if (has\_calculation && has\_binary) {

std::cout << " PASSED" << std::endl;

return true;

} else {

std::cout << " FAILED: Не знайдено очікувані результати" << std::endl;

return false;

}

}

// Функція для запуску всіх тестів

void runAllTests() {

std::cout << "=== ЗАПУСК АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ===" << std::endl;

std::cout << "Дата початку тестування: " << getCurrentDate() << std::endl << std::endl;

std::vector<std::pair<std::string, std::function<bool()>>> tests = {

{"Задача 10.1 - Кейс 1", test\_10\_1\_case\_1},

{"Задача 10.1 - Кейс 2", test\_10\_1\_case\_2},

{"Задача 10.2 - Кейс 1", test\_10\_2\_case\_1},

{"Задача 10.3 - Кейс 1", test\_10\_3\_case\_1}

};

int passed = 0;

int total = tests.size();

for (const auto& test : tests) {

std::cout << "Тестування: " << test.first << std::endl;

if (test.second()) {

passed++;

}

std::cout << std::endl;

}

std::cout << "=== РЕЗУЛЬТАТИ ТЕСТУВАННЯ ===" << std::endl;

std::cout << "Пройдено тестів: " << passed << "/" << total << std::endl;

std::cout << "Відсоток успішності: " << (passed \* 100.0 / total) << "%" << std::endl;

if (passed == total) {

std::cout << "? Всі тести пройдено успішно!" << std::endl;

} else {

std::cout << "? Деякі тести не пройшли перевірку." << std::endl;

}

}

int main() {

system("chcp 1251 && cls");

std::cout << "ТЕСТОВИЙ ДРАЙВЕР для лабораторної роботи №10" << std::endl;

std::cout << "===========================================" << std::endl;

runAllTests();

std::cout << std::endl << "Натисніть Enter для завершення...";

std::cin.get();

return 0;

}

# Аргументи досягнення мети

1. **Опанування синтаксису оголошення масивів**: Успішно засвоєно синтаксис оголошення одно- та двовимірних масивів у С/С++, що є фундаментальною основою для роботи зі складними структурами даних.
2. **Розуміння індексації масивів**: Отримано чітке розуміння принципів доступу до елементів масиву через індекси, що забезпечує ефективну роботу з великими обсягами даних.
3. **Практичне застосування масивів**: Реалізовано практичні навички роботи з масивами при розв'язуванні конкретних задач 10.1-10.3.
4. **Освоєння бібліотеки cstring**: Детально вивчено призначення та використання ключових функцій роботи з рядками (strstr, strlen, strcpy, strcmp та інші).
5. **Ефективна обробка текстових даних**: Набуто практичних навичок використання рядкових функцій для обробки символьної інформації в реальних задачах.
6. **Безпечна робота з рядками**: Засвоєно принципи використання безпечних версій функцій (strncpy, strncat) для запобігання переповненню буферів.
7. **Розуміння концепції структур**: Отримано глибоке розуміння призначення та можливостей структур (struct) як засобу організації складних даних.
8. **Практичне створення структур**: Успішно реалізовано оголошення та використання структур із різними типами полів.
9. **Робота з вкладеними структурами**: Опановано техніку створення структур, що містять інші структури як поля.
10. **Доступ до членів структур**: Засвоєно використання оператора доступу до полів структури для ефективної роботи з даними.
11. **Розуміння дескрипторів структур**: Отримано чітке уявлення про правила утворення та використання дескрипторів структур.
12. **Робота з множинними типами**: Успішно опановано синтаксис оголошення змінних множинного типу та допустимі операції над ними.
13. **Перетворення типів**: Засвоєно принципи явного та неявного перетворення типів даних у С/С++.
14. **Освоєння класів файлових потоків**: Детально вивчено роботу з класами fstream, ofstream, ifstream для різних режимів доступу до файлів.
15. **Режими відкриття файлів**: Опановано використання констант режимів відкриття (ios\_base::app, ios\_base::in, ios\_base::out тощо).
16. **Функції-члени потокових об'єктів**: Засвоєно призначення та використання функцій open(), eof(), close() для управління файловими потоками.
17. **Модульна архітектура**: Успішно реалізовано модульний підхід до розробки програмного забезпечення через створення статичної бібліотеки.
18. **Розділення інтерфейсу та реалізації**: Опановано принципи створення заголовкових файлів та їх використання для підключення функціональності.
19. **Повторне використання коду**: Досягнуто ефективного повторного використання розроблених модулів у різних проектах.
20. **Розробка тест-сьютів**: Створено комплексні набори тестових даних для автоматизованої перевірки функціональності модулів.
21. **Автоматизоване unit-тестування**: Реалізовано тестовий драйвер для автоматичної перевірки коректності роботи програмних модулів.
22. **Протоколювання тестування**: Впроваджено систему логування результатів тестування з детальною інформацією про статус кожного тест-кейса.
23. **Структурований підхід до тестування**: Застосовано професійну методологію тестування з використанням етапів Preliminary Steps, Action, Expected Result.
24. **Система контролю версій**: Опановано роботу з Git для управління версіями коду та забезпечення збереженості розробок.
25. **Організація файлової структури**: Створено логічну структуру проекту з окремими теками для різних типів артефактів.
26. **Документування проекту**: Реалізовано підтримку файлу README.md для опису проекту та інструкцій з використання.
27. **Інтеграція бібліотек**: Успішно налаштовано підключення створеної статичної бібліотеки до тестового драйвера.
28. **Налаштування середовища розробки**: Опановано конфігурацію Code::Blocks IDE для роботи з бібліотеками та заголовковими файлами.
29. **Кросплатформна сумісність**: Забезпечено сумісність розробленого ПЗ з різними операційними системами (Windows, Linux, Mac OS).
30. **Обробка файлових операцій**: Реалізовано перевірку статусу відкриття файлів та обробку можливих помилок.
31. **Відлагодження коду**: Набуто практичних навичок пошуку та усунення помилок у програмному коді.
32. **Валідація вхідних даних**: Впроваджено перевірку коректності вхідних параметрів функцій.
33. **Створення технічної документації**: Розроблено детальний звіт згідно з ДСТУ 3008:2015 з повним описом процесу розробки.
34. **Документування архітектури**: Створено артефакти проектування з детальним описом архітектури програмних модулів.
35. **Документування тестування**: Підготовлено повну документацію результатів тестування з аналізом отриманих результатів.
36. **Декомпозиція складних задач**: Успішно розбито складні задачі на менші підзадачі для ефективного розв'язування.
37. **Оптимізація алгоритмів**: Застосовано ефективні алгоритми для обробки даних різних типів.
38. **Аналіз складності**: Проведено аналіз часової та просторової складності реалізованих алгоритмів.
39. **Стиль кодування**: Дотримано професійних стандартів написання коду з належним форматуванням та коментуванням.
40. **Використання стандартних бібліотек**: Ефективно застосовано функції стандартних бібліотек С++ для розв'язування задач.
41. **Оптимізація пам'яті**: Реалізовано ефективне використання пам'яті при роботі зі складними структурами даних.
42. **Планування розробки**: Застосовано системний підхід до планування та виконання етапів розробки ПЗ.
43. **Контроль якості**: Впроваджено багаторівневу систему контролю якості коду через тестування та код-рев'ю.
44. **Управління конфігурацією**: Забезпечено правильне управління залежностями та конфігурацією проекту.
45. **Розуміння принципів ООП**: Застосовано об'єктно-орієнтовані принципи при роботі з потоковими об'єктами.
46. **Знання архітектурних патернів**: Використано патерни проектування для створення масштабованої архітектури.
47. **Розуміння системного програмування**: Отримано знання про взаємодію програм з файловою системою та операційною системою.
48. **Готовність до командної роботи**: Набуто навичок, необхідних для роботи в команді розробників через використання Git та стандартизованих підходів.
49. **Підготовка до промислової розробки**: Опановано інструменти та методології, які широко використовуються в промисловій розробці ПЗ.
50. **Формування професійного мислення**: Розвинуто системний підхід до розв'язування програмістських задач з урахуванням вимог якості, надійності та підтримуваності коду.