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 I/O

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 КІ - 24

Савельєв В. В.

ПЕРЕВІРИВ

ст. викладач кафедри кібербезпеки   
та програмного забезпечення

Коваленко А.С

Кропивницький – 2025

**ТЕМА:** Реалізація програмних модулів оброблення даних складових типів з файловим I/O

**МЕТА** Полягає у набутті ґрунтовних вмінь і практичних навичок реалізації у Code::Blocks IDE мовою 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 №10.1**

У вихідний текстовий файл записати:

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

кількість абзаців у тексті із вхідного файла;

повідомлення, чи є у тексті з вхідного файла слова "Україна", "університет" та "блокнот" (у програмі слід реалізувати розрізнення слів, наприклад, "університет" та "університету" тощо).

**АНАЛІЗ ЗАДАЧІ:**Необхідно опрацювати вхідний текстовий файл, порахувати кількість абзаців (які зазвичай відокремлені порожнім рядком або переходом на новий рядок) і визначити, чи містить текст точно слова "Україна", "університет" та "блокнот", а не їхні форми (тобто потрібно точне співпадіння, з урахуванням розділових знаків і пробілів). Потім результати разом з авторською інформацією слід записати у вихідний текстовий файл.

**ПРОЕКТНЕ РІШЕННЯ:**Програма читає вміст вхідного файлу, розбиває його на абзаци, рахує їхню кількість. Для перевірки слів використовується розбиття тексту на окремі слова з урахуванням розділових знаків, щоб знайти точне співпадіння. Далі створюється новий текстовий файл, у який записується авторська інформація, кількість абзаців та повідомлення про наявність кожного з трьох слів**.**

**РЕАЛІЗАЦІЯ БІБЛІОТЕКИ**

Вихідний файл ModulesCovelyev.cpp

void process\_text\_file(const string& input\_filename, const string& output\_filename,

const string& author\_name, const string& institution,

const string& city, const string& country, int year) {

ifstream in\_file(input\_filename);

ofstream out\_file(output\_filename);

if (!in\_file.is\_open() || !out\_file.is\_open()) {

cerr << "Помилка: неможливо відкрити файли!" << endl;

return;

}

out\_file << "Автор: " << author\_name << "\n";

out\_file << "Установа: " << institution << "\n";

out\_file << "Місто: " << city << "\n";

out\_file << "Країна: " << country << "\n";

out\_file << "Рік: " << year << "\n\n";

string line;

int paragraph\_count = 0;

bool has\_ukraine = false, has\_university = false, has\_notebook = false;

bool in\_paragraph = false;

string ukraine\_words[] = { "україна", "україни", "україні", "україну", "Україна", "України", "Україні", "Україну" };

string university\_words[] = { "університет", "університету", "університети", "університетів", "Університет", "Університету", "Університети", "Університетів" };

string notebook\_words[] = { "блокнот", "блокнота", "блокноту", "блокноти", "Блокнот", "Блокнота", "Блокноту", "Блокноти" };

while (getline(in\_file, line)) {

if (line.empty()) {

if (in\_paragraph) {

paragraph\_count++;

in\_paragraph = false;

}

}

else {

in\_paragraph = true;

for (const string& word : ukraine\_words) {

if (!has\_ukraine && line.find(word) != string::npos) {

has\_ukraine = true;

}

}

for (const string& word : university\_words) {

if (!has\_university && line.find(word) != string::npos) {

has\_university = true;

}

}

for (const string& word : notebook\_words) {

if (!has\_notebook && line.find(word) != string::npos) {

has\_notebook = true;

}

}

}

}

if (in\_paragraph) {

paragraph\_count++;

}

out\_file << "Кількість абзаців: " << paragraph\_count << "\n";

out\_file << "Наявність слів:\n";

out\_file << "Україна: " << (has\_ukraine ? "так" : "ні") << "\n";

out\_file << "Університет: " << (has\_university ? "так" : "ні") << "\n";

out\_file << "Блокнот: " << (has\_notebook ? "так" : "ні") << "\n";

}

Заголовковий файл ModulesCovelyev.h

void process\_text\_file(const string& input\_filename, const string& output\_filename,

const string& author\_name, const string& institution,

const string& city, const string& country, int year);

**РЕЗУЛЬТАТ ВИКОНАННЯ ТЕСТОВОГО ДРАЙВЕРА:**

Тест кейс 1 (Basic Functionality) | Вхід: "Це перший абзац про те що Україна славна та непереможна.

Другий абзац розповідає про університет у місті Київ.

Третій абзац згадує блокнот, який я купив учора.

"

Очікуваний результат:

Автор: Володимир Савельєв

Установа: ЦНТУ

Місто: Кропивницький

Країна: Україна

Рік: 2025

Кількість абзаців: 3

Наявність слів:

Україна: так

Університет: так

Блокнот: так

Фактичний результат:

Автор: Володимир Савельєв

Установа: ЦНТУ

Місто: Кропивницький

Країна: Україна

Рік: 2025

Кількість абзаців: 3

Наявність слів:

Україна: так

Університет: так

Блокнот: так

Результат: PASSED

--------------------------------------

Тест кейс 2 (No Keywords) | Вхід: "Це текст без ключових слів.

"

Очікуваний результат:

Автор: Володимир Савельєв

Установа: ЦНТУ

Місто: Кропивницький

Країна: Україна

Рік: 2025

Кількість абзаців: 1

Наявність слів:

Україна: ні

Університет: ні

Блокнот: ні

Фактичний результат:

Автор: Володимир Савельєв

Установа: ЦНТУ

Місто: Кропивницький

Країна: Україна

Рік: 2025

Кількість абзаців: 1

Наявність слів:

Україна: ні

Університет: ні

Блокнот: ні

Результат: PASSED

--------------------------------------

Тест кейс 3 (Empty File) | Вхід: ""

Очікуваний результат:

Автор: Володимир Савельєв

Установа: ЦНТУ

Місто: Кропивницький

Країна: Україна

Рік: 2025

Кількість абзаців: 0

Наявність слів:

Україна: ні

Університет: ні

Блокнот: ні

Фактичний результат:

Автор: Володимир Савельєв

Установа: ЦНТУ

Місто: Кропивницький

Країна: Україна

Рік: 2025

Кількість абзаців: 0

Наявність слів:

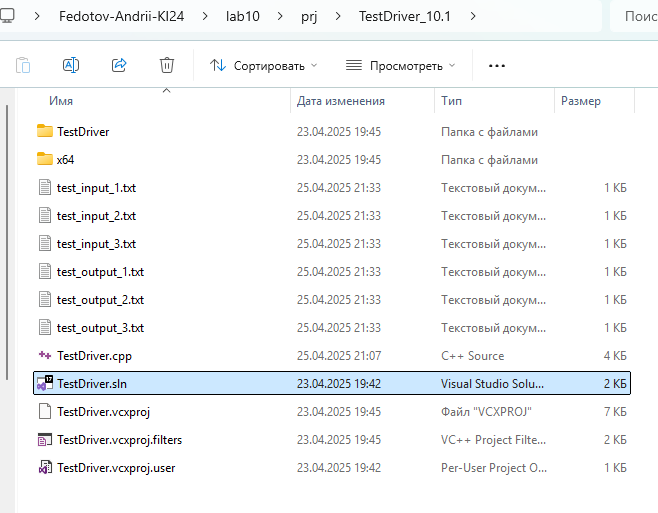
Україна: ні

Університет: ні

Блокнот: ні

Результат: PASSED

--------------------------------------



Малюнок 1 - Файли програми після запуску

**ЗАВДАННЯ №10.2**

У вхідний текстовий файл дописати:

кількість знаків пунктуації у ньому, дату й час дозапису інформації.

**АНАЛІЗ ЗАДАЧІ:**Потрібно до вхідного текстового файлу (той самий, що і в попередньому завданні) дописати кількість знаків пунктуації, а також поточну дату та час. Це означає, що файл не перезаписується, а доповнюється вмістом.

**ПРОЕКТНЕ РІШЕННЯ:**Програма відкриває вхідний файл для читання, підраховує кількість символів пунктуації (.,!?;:- тощо), потім відкриває файл для дозапису та дописує знайдену кількість, а також дату та час у форматі, наприклад, "07.05.2025 14:20".

**РЕАЛІЗАЦІЯ БІБЛІОТЕКИ**

Вихідний файл ModulesCovelyev.cpp

void append\_punctuation\_count(const string& input\_filename, const string& output\_filename) {

ifstream in\_file(input\_filename);

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

ofstream error\_file(output\_filename);

error\_file << "Помилка: неможливо відкрити файл!\n";

error\_file.close();

return;

}

stringstream buffer;

buffer << in\_file.rdbuf();

string content = buffer.str();

in\_file.close();

// Видаляємо \r для коректної роботи на Windows

content.erase(remove(content.begin(), content.end(), '\r'), content.end());

int punctuation\_count = count\_if(content.begin(), content.end(), [](char c) {

return c == '.' || c == ',' || c == '!' || c == '?' || c == ';' || c == ':';

});

ofstream out\_file(output\_filename);

out\_file << "Кількість знаків пунктуації: " << punctuation\_count << "\n";

time\_t now = time(nullptr);

tm local\_time{};

#ifdef \_WIN32

localtime\_s(&local\_time, &now);

#else

local\_time = \*localtime(&now);

#endif

out\_file << "Дата та час запису: " << put\_time(&local\_time, "%Y-%m-%d %H:%M:%S") << "\n";

out\_file.close();

}

Заголовковий файл ModulesCovelyev.h

void append\_punctuation\_count(const string&, const string&);

**РЕЗУЛЬТАТ ВИКОНАННЯ ТЕСТОВОГО ДРАЙВЕРА:**

Тест кейс 1 | Вхідний файл: test\_input\_1.txt | Вихідний файл: test\_output\_1.txt

Вхідний вміст:

Це текст, із кількома знаками! Пунктуація: так, ще.

Другий рядок; із комами,, і крапками.

Очікуваний вміст (префікс):

Кількість знаків пунктуації: 9

<дата та час>

Фактичний вміст:

Кількість знаків пунктуації: 9

Дата та час запису: 2025-05-07 17:16:55

Результат: PASSED

--------------------------------------

Тест кейс 2 | Вхідний файл: test\_input\_2.txt | Вихідний файл: test\_output\_2.txt

Вхідний вміст:

Це текст без пунктуації

Другий рядок також без знаків

Очікуваний вміст (префікс):

Кількість знаків пунктуації: 0

<дата та час>

Фактичний вміст:

Кількість знаків пунктуації: 0

Дата та час запису: 2025-05-07 17:16:55

Результат: PASSED

--------------------------------------

Тест кейс 3 | Вхідний файл: test\_input\_3.txt | Вихідний файл: test\_output\_3.txt

Вхідний вміст:

Це текст, із кількома знаками! Пунктуація: так, ще.

Другий рядок; із комами, і крапками, ще.

Очікуваний вміст (префікс):

Кількість знаків пунктуації: 9

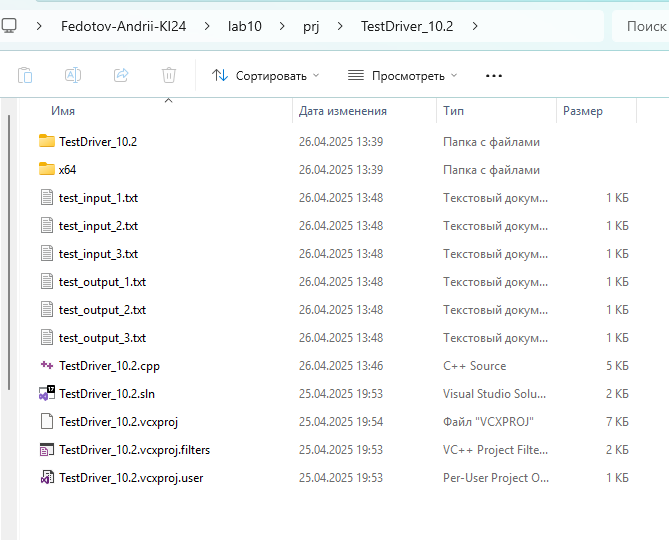
<дата та час>

Фактичний вміст:

Кількість знаків пунктуації: 9

Дата та час запису: 2025-05-07 17:16:55

Результат: PASSED



Малюнок 2 - Файли програми після запуску

**ЗАВДАННЯ №10.3**

Вхідні дані числові значення х, у, та натуральне число в. У вихідний текстовий файл дописати:

результати виконання функцій із заголовкового файлу Modules Прізвище.һ

s\_calculation з аргументами х, у, г; число в у двійковому коді.

**АНАЛІЗ ЗАДАЧІ:**Задані три числа: два дійсних числа х і у, і натуральне число в. Потрібно обчислити значення деякої функції s\_calculation (яка реалізована в окремому заголовковому файлі Modules Прізвище.һ) з цими аргументами, а також представити число в у вигляді двійкового коду. Результати потрібно записати у вихідний текстовий файл.

**ПРОЕКТНЕ РІШЕННЯ:**У програмі підключається заголовковий файл з функцією s\_calculation. Користувач вводить значення х, у, z та b. Викликається функція з цими аргументами, а також виводиться двійкове представлення числа в за допомогою побітових операцій або стандартних засобів. Отримані результати дозаписуються у вихідний текстовий файл.

**РЕАЛІЗАЦІЯ БІБЛІОТЕКИ**

Вихідний файл ModulesCovelyev.cpp

void append\_calculation\_results(const string& output\_filename, double x, double y, double a, unsigned int b) {

ofstream out\_file(output\_filename, ios::app);

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

cerr << "Помилка: неможливо відкрити файл!\n";

return;

}

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

out\_file << fixed << setprecision(6);

out\_file << "\nРезультат s\_calculation(" << x << ", " << y << ", " << a << "): " << result << "\n";

string binary = b == 0 ? "0" : "";

unsigned int temp = b;

while (temp > 0) {

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

temp /= 2;

}

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

}

Заголовковий файл ModulesCovelyev.h

void append\_calculation\_results(const string& output\_filename, double x, double y, double a, unsigned int b);

**РЕЗУЛЬТАТ ВИКОНАННЯ ТЕСТОВОГО ДРАЙВЕРА:**

Тест кейс 1 | Вхідний файл: test\_input\_1.txt | Вихідний файл: test\_output\_1.txt

Вхідні дані: x=1, y=2, z=3, b=5

Очікуваний вміст:

Результат s\_calculation(1.000000, 2.000000, 3.000000): 8.265324

Число 5 у двійковому коді: 101

Фактичний вміст:

Результат s\_calculation(1.000000, 2.000000, 3.000000): 8.265324

Число 5 у двійковому коді: 101

Результат: PASSED

--------------------------------------

Тест кейс 2 | Вхідний файл: test\_input\_2.txt | Вихідний файл: test\_output\_2.txt

Вхідні дані: x=-1.5, y=0.5, z=-2, b=0

Очікуваний вміст:

Результат s\_calculation(-1.500000, 0.500000, -2.000000): -5.127335

Число 0 у двійковому коді: 0

Фактичний вміст:

Результат s\_calculation(-1.500000, 0.500000, -2.000000): -5.127335

Число 0 у двійковому коді: 0

Результат: PASSED

--------------------------------------

Тест кейс 3 | Вхідний файл: test\_input\_3.txt | Вихідний файл: test\_output\_3.txt

Вхідні дані: x=0.75, y=1.25, z=1.5, b=13

Очікуваний вміст:

Результат s\_calculation(0.750000, 1.250000, 1.500000): 5.863084

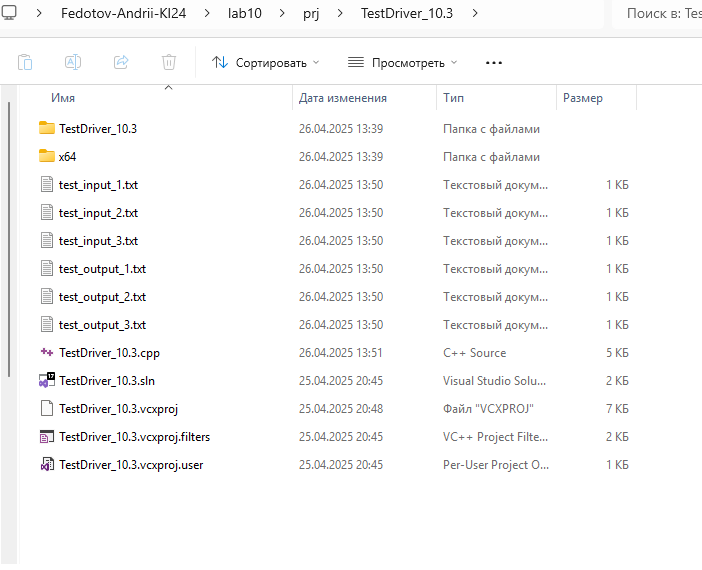
Число 13 у двійковому коді: 1101

Фактичний вміст:

Результат s\_calculation(0.750000, 1.250000, 1.500000): 5.863084

Число 13 у двійковому коді: 1101

Результат: PASSED



Малюнок 3 - Файли програми після запуску

**ВІДПОВІДЬ НА КОНТРОЛЬНІ ЗАПИТАННЯ**

1. Блок try-throw-catch у C++ призначений для обробки винятків, тобто непередбачуваних ситуацій під час виконання програми. Синтаксис: у блоці try розміщується код, що потенційно може згенерувати помилку, throw використовується для створення винятку, а catch — для його перехоплення і обробки.
2. Міжмодульна змінна — це глобальна змінна, яка оголошується в одному файлі за допомогою ключового слова extern у заголовковому файлі, а визначається в одному з модулів програми. Це дозволяє спільно використовувати її в кількох модулях.
3. Об’єкти, описані в тілі функції main, мають локальну область видимості, тобто доступні лише всередині цієї функції та недоступні за її межами.
4. Змінна типу enum описує перелік логічно пов’язаних констант із власними іменами, тоді як масив зберігає набір значень одного типу. enum забезпечує читабельність і зручність при роботі з фіксованими варіантами, тоді як масив дозволяє зберігати та обробляти множину значень.
5. Потік — це абстракція джерела або приймача даних. Файловий потік — це потік, що взаємодіє з файлами на диску. Він відрізняється від стандартних потоків тим, що дозволяє читати/записувати дані з/у зовнішні файли, а не з клавіатури чи на екран.
6. Універсальний алгоритм: відкрити файл, перевірити його успішність, зчитати або записати дані, закрити файл. Можна використовувати об'єкти ifstream, ofstream або fstream.
7. Для запису числа π з точністю до 106 цифр потрібно як мінімум 107 символів (включно з "3."), отже обсяг файла становитиме приблизно 107 байтів, якщо кодування ASCII або UTF-8 без BOM.
8. Перевірка здійснюється через метод is\_open() або перевірку стану потоку. Її мета — упевнитися, що файл відкрився успішно, інакше уникнути аварійного завершення програми.
9. Текстовий потік оперує символами і форматованими даними, тоді як двійковий — байтами без інтерпретації. У двійковому потоці дані читаються і записуються в тому вигляді, в якому вони є в пам’яті.
10. Режими відкриття: читання, запис, додавання, бінарний режим, перезапис, апенд (дозапис у кінець), комбіновані варіанти для читання і запису.
11. У fstream реалізовано класи ifstream для читання, ofstream для запису, fstream для читання і запису одночасно. Для створення об'єктів використовуються конструкції типу ifstream in("file.txt"); і ofstream out("file.txt");.
12. Константи: ios::in — відкриття для читання, ios::out — для запису, ios::app — дозапис у кінець, ios::binary — двійковий режим, ios::trunc — обтинання вмісту, ios::ate — відкриття з переходом у кінець файла.
13. Потоки відкриваються методом open(), закриваються методом close(), кінець файлу визначається функцією eof().
14. Алгоритм: прочитати файл у рядок або список рядків, видалити всі входження послідовності (слова), записати результат у файл знову, замінивши старий вміст або створивши новий файл.
15. Складові типи: масиви, структури, об’єднання, класи. Вони дозволяють об’єднувати кілька значень під одним ідентифікатором, мають власні правила ініціалізації і доступу до елементів, складніші за прості типи.
16. У C/C++ множину реалізують через структури даних, наприклад, масиви або std::set. Перевірка — через пошук, додавання — через вставку, вилучення — через видалення.
17. Явне перетворення (casting) здійснюється через оператор (тип), неявне — відбувається автоматично, коли компілятор може безпечно змінити тип (наприклад, int → float).
18. union доцільно використовувати, коли потрібно економити пам’ять і зберігати в одному об'єкті лише одне з кількох можливих значень. Наприклад, один об'єкт може зберігати або ціле, або дійсне число залежно від контексту.
19. Алгоритм: зчитати вміст файла, перевірити за допомогою пошуку або регулярного виразу наявність заданого числа, повідомити про результат.
20. Вираз sizeof short('R') некоректний, бо short — не функція. Правильний варіант — sizeof(short), який дорівнює 2 байти на більшості систем. Змінна symbol\_transaction отримає значення 4 або помилку компіляції, залежно від компілятора.

**ВИСНОВКИ**

1. Опрацьовано основи роботи з масивами в мові C++.
2. Освоєно створення та заповнення структур даних.
3. Вивчено принципи використання об'єднань (union).
4. Засвоєно роботу з типом даних "перелік" (enum).
5. Розглянуто поняття множин у програмуванні.
6. Вивчено способи перетворення типів даних.
7. Набуто навичок роботи з символами.
8. Зрозуміло основи роботи з файлами для введення та виведення.
9. Опрацьовано відкриття та закриття файлів.
10. Освоєно методи обробки текстових файлів.
11. Використано функції для збереження інформації у файли.
12. Навченося зчитувати дані з файлів різних форматів.
13. Вивчено способи обробки помилок при роботі з файлами.
14. Освоєно використання динамічних масивів.
15. Вивчено роботу з рядками за допомогою бібліотеки <string>.
16. Засвоєно принципи модульної організації програмного коду.
17. Розвинуто логічне мислення та аналітичні навички.
18. Набуто досвіду у створенні користувацьких функцій.
19. Вивчено передавання параметрів до функцій.
20. Практиковано повернення значень з функцій.
21. Освоєно роботу з заголовковими файлами.
22. Засвоєно інкапсуляцію даних у структурах.
23. Вивчено техніки форматованого виводу.
24. Реалізовано умовні конструкції на практиці.
25. Застосовано цикли для обробки різних даних.
26. Вивчено вкладені цикли в роботі з масивами.
27. Освоєно методи сортування масивів.
28. Практиковано пошук елементів у масиві.
29. Розглянуто копіювання та порівняння рядків.
30. Отримано навички у відлагодженні програм.
31. Освоєно роботу з середовищем Code::Blocks.
32. Вивчено процес компіляції та запуску програм.
33. Навченося структурувати код за окремими файлами.
34. Отримано досвід у створенні власних заголовкових файлів.
35. Практиковано інтеграцію кількох файлів у межах одного проєкту.
36. Засвоєно використання покажчиків у поєднанні з масивами.
37. Вивчено передачу структур як параметрів функцій.
38. Освоєно роботу з символьними потоками.
39. Розширено знання про типи даних у C++.
40. Практично вивчено оголошення змінних різних типів.
41. Засвоєно значення ініціалізації змінних.
42. Вивчено як автоматичне, так і явне перетворення типів.
43. Набуто досвіду роботи з typedef і enum.
44. Застосовано об'єднання типів на практиці.
45. Засвоєно структуру циклів for, while, do-while.
46. Освоєно умовні оператори if та switch.
47. Вивчено роботу з бібліотеками <fstream>, <iostream>, <cstring>.
48. Практиковано використання логічних операторів.
49. Отримано досвід у програмній обробці даних.
50. Поглиблено розуміння концепцій модульного програмування.