Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

**Звіт**

Лабораторна робота №12

**ТЕМА: Програмна реалізація абстрактних типів даних**

Виконав:cт.гр.КБ-24

Ковальчук В.О.

Перевірив: викладач

О. П. Доренський

**ТЕМА: Програмна реалізація абстрактних типів даних**

**Мета : Полягає у набутті ґрунтовних вмінь і практичних**

**навичок об’єктного аналізу й проєктування, створення класів С++ та тестування їх екземплярів, використання препроцесорних директив, макросів і макрооператорів під час реалізації програмних засобів у кросплатформовому середовищі Code::Blocks.**

**Варіант 38**

**Завдання :**

1. Як складову заголовкового файлу ModulesПрізвище.h розробити

клас ClassLab12\_Прізвище –– формальне представлення

абстракції сутності предметної області (об’єкта) за варіантом, ―

поведінка об’єкта якого реалізовує розв’язування задачі 12.1.

2. Реалізувати додаток Teacher, який видає 100 звукових сигналів і в

текстовий файл TestResults.txt записує рядок “Встановлені вимоги

порядку виконання лабораторної роботи порушено!”, якщо файл

проєкта main.срр під час його компіляції знаходився не в \Lab12\prj,

інакше –– створює об’єкт класу ClassLab12\_Прізвище із

заголовкового файлу ModulesПрізвище.h та виконує його unit-тестування

за тест-сьютом(ами) із \Lab12\TestSuite\, протоколюючи результати

тестування в текстовий файл \Lab12\TestSuite\TestResults.txt.

**Дано наступну сутність предметної області (об'єкт).**

Об'єкт (екземпляр) класу ClassLab12\_Премиице, як абстракція даної сутності предметної області, за наданим інтерфейсом забезпечує:

* надання значень своїх атрибутів,
* надання значения площі поверхні столу
* зміну значения заданого атрибута(ів).

Під час створення об'ектаклесу вся його атрибути зніцшалізуються конструктором.

2 Під наданням розуміється повернення результату відповідними функциями-членами об'екта класу.

Площа обчислюється і повернеться відкидаю функцию-сеном (методом) об'екта касу значеннями його играбув

Всі дані-члени класу е закрепа (private); доступ до них (читання, запие) реалізують відповідні

**Призначення програми**

Програма реалізує роботу з класом еліптичного столу ClassLab12\_Kovalchuk:

* Створює об’єкти столів (довжина й ширина — як півосі еліпса);
* Обчислює площу столу;
* Порівнює площі двох столів;
* Проводить модульне тестування;
* Перевіряє, чи виконання відбувається у правильній директорії;
* демонструє роботу класу.

Загальна логіка виконання (main):

1. Налаштовує українську мову у консолі (setupUkrainianConsole).

2. Виводить заголовок лабораторної.

3. Перевіряє директорію запуску (чи містить lab12 і prj).

Якщо не відповідає — запитує користувача або завершує роботу з сигналом (beep100times).

4. Створює папку TestSuite, якщо її ще нема.

5. Виконує модульні тести на правильність роботи класу (функція

runTests).

6. Демонструє функціонал класу (demonstrateClass): створення, зміна

параметрів, порівняння площ.

7. Завершує програму з повідомленням.

**Окремі компоненти**

Клас ClassLab12\_Kovalchuk (реалізація в іншому файлі — не надано, але з прикладів видно):

* Має параметри a, b (напівосі еліпса);
* Методи: getA(), getB() — отримання значень; setA(), setB() —

встановлення;

* calculateArea() — обчислення площі за формулою ;
* displayInfo() — вивід параметрів у консоль;
* isLargerThan() — порівняння площ столів; перевантаження << для зручного виводу.

**Модульне тестування (runTests)**

**Перевіряються:**

* Правильність ініціалізації;
* Обчислення площ;
* Зміна параметрів;
* Граничні значення;
* Виводиться звіт у файл.

**Файлова система**

Програма очікує, що її запустять із шляху на кшталт:

*...\Lab12\prj\*

* Результати зберігає у файл TestResults.txt у директорії:
* ../TestSuite/, або
* ./TestSuite/, або
* Поточній директорії — якщо інші варіанти недоступні.

**Повідомлення та контроль**

* Якщо структура папок невірна — Beep 100 разів;
* Якщо все успішно — виводиться підсумок.

**Завдання :**

*#include <iostream>*

*#include <fstream>*

*#include <Windows.h>*

*#include <locale.h>*

*#include <iomanip>*

*#include <string>*

*#include <cmath>*

*#include "ModulseKovalchuk.h"*

*using namespace std;*

*// Функція для звукового сигналу при порушенні вимог*

*void beep100times() {*

*cout << "УВАГА: Порушено вимоги виконання лабораторної роботи!" << endl;*

*for (int i = 0; i < 100; ++i) {*

*Beep(750, 100); // 750 Гц, 100 мс*

*if (i % 10 == 0) {*

*cout << "Beep " << i + 1 << "/100" << endl;*

*}*

*}*

*}*

*// Покращена перевірка коректності директорії*

*bool isInCorrectDirectory() {*

*char buffer[MAX\_PATH];*

*DWORD length = GetCurrentDirectoryA(MAX\_PATH, buffer);*

*if (length == 0 || length > MAX\_PATH) {*

*cerr << "Помилка при отриманні поточної директорії." << endl;*

*return false;*

*}*

*string path(buffer);*

*cout << "Поточна директорія: " << path << endl;*

*// Конвертуємо в нижній регістр для порівняння*

*string lowerPath = path;*

*for (char& c : lowerPath) {*

*c = tolower(c);*

*}*

*// Перевіряємо різні варіанти назв*

*bool hasLab12 = (lowerPath.find("lab12") != string::npos ||*

*lowerPath.find("лаб12") != string::npos ||*

*lowerPath.find("lab\_12") != string::npos);*

*bool hasPrj = (lowerPath.find("prj") != string::npos ||*

*lowerPath.find("proj") != string::npos ||*

*lowerPath.find("project") != string::npos ||*

*lowerPath.find("проект") != string::npos);*

*cout << "Перевірка структури:" << endl;*

*cout << "- Містить Lab12: " << (hasLab12 ? "так" : "ні") << endl;*

*cout << "- Містить prj: " << (hasPrj ? "так" : "ні") << endl;*

*// Якщо не знайдено обидва компоненти, запитуємо користувача*

*if (!hasLab12 || !hasPrj) {*

*cout << "\nУВАГА: Автоматична перевірка не знайшла очікувану структуру." << endl;*

*cout << "Чи бажаєте продовжити виконання програми? (y/n): ";*

*char choice;*

*cin >> choice;*

*cin.ignore(); // очищаємо буфер*

*if (choice == 'y' || choice == 'Y' || choice == 'т' || choice == 'Т') {*

*cout << "Продовжуємо виконання за вашим вибором..." << endl;*

*return true;*

*} else {*

*return false;*

*}*

*}*

*return true;*

*}*

*// Створення директорії TestSuite*

*void ensureTestSuiteDirectory() {*

*// Спробуємо різні варіанти створення директорії*

*const char\* testDirs[] = {*

*"..*[*\\TestSuite*](file://\\testsuite\)*",*

*".*[*\\TestSuite*](file://\\testsuite\)*",*

*"TestSuite"*

*};*

*bool created = false;*

*for (const char\* dir : testDirs) {*

*if (CreateDirectoryA(dir, NULL)) {*

*cout << "Створено директорію: " << dir << endl;*

*created = true;*

*break;*

*} else if (GetLastError() == ERROR\_ALREADY\_EXISTS) {*

*cout << "Директорія " << dir << " вже існує." << endl;*

*created = true;*

*break;*

*}*

*}*

*if (!created) {*

*cout << "Попередження: Не вдалося створити директорію TestSuite." << endl;*

*cout << "Результати будуть збережені в поточній директорії." << endl;*

*}*

*}*

*// Розширена функція тестування*

*void runTests(const string& resultPath) {*

*ofstream fout(resultPath);*

*if (!fout.is\_open()) {*

*// Якщо не можемо створити файл за вказаним шляхом, створимо в поточній директорії*

*string localPath = "TestResults.txt";*

*fout.open(localPath);*

*if (!fout.is\_open()) {*

*cerr << "Не вдалося відкрити файл для запису результатів тестів." << endl;*

*return;*

*}*

*cout << "Результати збережено в поточній директорії: " << localPath << endl;*

*}*

*cout << "Виконання модульного тестування..." << endl;*

*fout << "============================================" << endl;*

*fout << " РЕЗУЛЬТАТИ МОДУЛЬНОГО ТЕСТУВАННЯ" << endl;*

*fout << " Модуль: ModulseKovalchuk.h" << endl;*

*fout << " Клас: ClassLab12\_Kovalchuk" << endl;*

*fout << "============================================" << endl;*

*fout << fixed << setprecision(6);*

*int passedTests = 0;*

*int totalTests = 0;*

*cout << "\n--- Початок тестування ---" << endl;*

*// Тест 1: Конструктор за замовчуванням*

*totalTests++;*

*fout << "\nТест 1: Конструктор за замовчуванням" << endl;*

*cout << "\nВиконується Тест 1..." << endl;*

*ClassLab12\_Kovalchuk obj1;*

*double expectedA1 = 5.0, expectedB1 = 3.0;*

*bool test1 = (obj1.getA() == expectedA1 && obj1.getB() == expectedB1);*

*fout << " Очікувано: a=" << expectedA1 << ", b=" << expectedB1 << endl;*

*fout << " Отримано: a=" << obj1.getA() << ", b=" << obj1.getB() << endl;*

*fout << " Результат: " << (test1 ? "ПРОЙДЕНО" : "НЕ ПРОЙДЕНО") << endl;*

*cout << "Тест 1: " << (test1 ? " ПРОЙДЕНО" : " НЕ ПРОЙДЕНО") << endl;*

*if (test1) passedTests++;*

*// Тест 2: Конструктор з параметрами*

*totalTests++;*

*fout << "\nТест 2: Конструктор з параметрами" << endl;*

*cout << "\nВиконується Тест 2..." << endl;*

*ClassLab12\_Kovalchuk obj2(3.0, 2.0);*

*bool test2 = (obj2.getA() == 3.0 && obj2.getB() == 2.0);*

*fout << " Очікувано: a=3.0, b=2.0" << endl;*

*fout << " Отримано: a=" << obj2.getA() << ", b=" << obj2.getB() << endl;*

*fout << " Результат: " << (test2 ? "ПРОЙДЕНО" : "НЕ ПРОЙДЕНО") << endl;*

*cout << "Тест 2: " << (test2 ? " ПРОЙДЕНО" : " НЕ ПРОЙДЕНО") << endl;*

*if (test2) passedTests++;*

*// Тест 3: Обчислення площі*

*totalTests++;*

*fout << "\nТест 3: Обчислення площі еліпса" << endl;*

*cout << "\nВиконується Тест 3..." << endl;*

*double expectedArea = M\_PI \* 3.0 \* 2.0;*

*double actualArea = obj2.calculateArea();*

*bool test3 = (abs(actualArea - expectedArea) < 1e-10);*

*fout << " Очікувано: " << expectedArea << ", Отримано: " << actualArea << endl;*

*fout << " Результат: " << (test3 ? "ПРОЙДЕНО" : "НЕ ПРОЙДЕНО") << endl;*

*cout << "Тест 3: " << (test3 ? " ПРОЙДЕНО" : " НЕ ПРОЙДЕНО") << endl;*

*if (test3) passedTests++;*

*// Тест 4: Методи встановлення*

*totalTests++;*

*fout << "\nТест 4: Методи setA() та setB()" << endl;*

*cout << "\nВиконується Тест 4..." << endl;*

*obj2.setA(4.0);*

*obj2.setB(5.0);*

*bool test4 = (obj2.getA() == 4.0 && obj2.getB() == 5.0);*

*fout << " Результат: " << (test4 ? "ПРОЙДЕНО" : "НЕ ПРОЙДЕНО") << endl;*

*cout << "Тест 4: " << (test4 ? " ПРОЙДЕНО" : " НЕ ПРОЙДЕНО") << endl;*

*if (test4) passedTests++;*

*// Тест 5: Площа після зміни параметрів*

*totalTests++;*

*fout << "\nТест 5: Площа після зміни параметрів" << endl;*

*cout << "\nВиконується Тест 5..." << endl;*

*double newArea = obj2.calculateArea();*

*double expectedNewArea = M\_PI \* 4.0 \* 5.0;*

*bool test5 = (abs(newArea - expectedNewArea) < 1e-10);*

*fout << " Очікувано: " << expectedNewArea << ", Отримано: " << newArea << endl;*

*fout << " Результат: " << (test5 ? "ПРОЙДЕНО" : "НЕ ПРОЙДЕНО") << endl;*

*cout << "Тест 5: " << (test5 ? " ПРОЙДЕНО" : " НЕ ПРОЙДЕНО") << endl;*

*if (test5) passedTests++;*

*// Тест 6: Граничні значення*

*totalTests++;*

*fout << "\nТест 6: Граничні значення" << endl;*

*cout << "\nВиконується Тест 6..." << endl;*

*ClassLab12\_Kovalchuk obj3(0.1, 0.1);*

*double area3 = obj3.calculateArea();*

*double expected3 = M\_PI \* 0.1 \* 0.1;*

*bool test6 = (abs(area3 - expected3) < 1e-10);*

*fout << " Очікувана площа: " << expected3 << ", Отримана площа: " << area3 << endl;*

*fout << " Результат: " << (test6 ? "ПРОЙДЕНО" : "НЕ ПРОЙДЕНО") << endl;*

*cout << "Тест 6: " << (test6 ? " ПРОЙДЕНО" : " НЕ ПРОЙДЕНО") << endl;*

*if (test6) passedTests++;*

*// Підсумок*

*fout << "\n============================================" << endl;*

*fout << " ПІДСУМКИ ТЕСТУВАННЯ" << endl;*

*fout << "============================================" << endl;*

*fout << "Загальна кількість тестів: " << totalTests << endl;*

*fout << "Пройдено тестів: " << passedTests << endl;*

*fout << "Не пройдено тестів: " << (totalTests - passedTests) << endl;*

*fout << "Відсоток успішності: " << (100.0 \* passedTests / totalTests) << "%" << endl;*

*if (passedTests == totalTests) {*

*fout << "\n ВСІ ТЕСТИ ПРОЙДЕНО УСПІШНО!" << endl;*

*cout << "\n ВСІ ТЕСТИ ПРОЙДЕНО УСПІШНО!" << endl;*

*} else {*

*fout << "\n ДЕЯКІ ТЕСТИ НЕ ПРОЙДЕНО!" << endl;*

*cout << "\n ДЕЯКІ ТЕСТИ НЕ ПРОЙДЕНО! Перевірте результати." << endl;*

*}*

*fout.close();*

*cout << "\nРезультати тестування збережено у файл: " << resultPath << endl;*

*}*

*// Демонстрація роботи класу*

*void demonstrateClass() {*

*cout << "\n=== ДЕМОНСТРАЦІЯ РОБОТИ КЛАСУ ===" << endl;*

*cout << "\n1. Створення столів:" << endl;*

*ClassLab12\_Kovalchuk table1; // за замовчуванням*

*ClassLab12\_Kovalchuk table2(6.0, 4.0); // з параметрами*

*cout << "\n2. Відображення інформації:" << endl;*

*cout << "\nСтіл 1 (за замовчуванням):" << endl;*

*table1.displayInfo();*

*cout << "Стіл 2 (з параметрами):" << endl;*

*table2.displayInfo();*

*cout << "\n3. Зміна параметрів столу 1:" << endl;*

*table1.setA(7.5);*

*table1.setB(5.2);*

*cout << "Оновлений стіл 1:" << endl;*

*table1.displayInfo();*

*cout << "\n4. Порівняння столів:" << endl;*

*if (table1.isLargerThan(table2)) {*

*cout << "Стіл 1 має більшу площу ніж стіл 2" << endl;*

*} else {*

*cout << "Стіл 2 має більшу або рівну площу зі столом 1" << endl;*

*}*

*cout << "\n5. Використання оператора виводу:" << endl;*

*cout << "Стіл 1: " << table1 << endl;*

*cout << "Стіл 2: " << table2 << endl;*

*}*

*// Головна функція*

*int main() {*

*// Налаштування кодування для української мови*

*SetConsoleCP(1251);*

*SetConsoleOutputCP(1251);*

*setlocale(LC\_ALL, "Ukrainian");*

*cout << "=========================================" << endl;*

*cout << " ЛАБОРАТОРНА РОБОТА 12" << endl;*

*cout << " Варіант 38" << endl;*

*cout << " Модуль: ModulseKovalchuk.h" << endl;*

*cout << " Клас: ClassLab12\_Kovalchuk" << endl;*

*cout << " Тема: Еліптичні столи" << endl;*

*cout << "=========================================" << endl;*

*string resultFile = "TestResults.txt";*

*// Перевірка структури директорій*

*if (!isInCorrectDirectory()) {*

*cout << "\n ПОМИЛКА: Структура директорій не відповідає вимогам!" << endl;*

*cout << "Або ви вирішили не продовжувати виконання." << endl;*

*// Звуковий сигнал про помилку*

*beep100times();*

*// Запис помилки у файл*

*ofstream fout(resultFile);*

*if (fout.is\_open()) {*

*fout << "ПОМИЛКА ВИКОНАННЯ ПРОГРАМИ" << endl;*

*fout << "Дата: " << \_\_DATE\_\_ << " " << \_\_TIME\_\_ << endl;*

*fout << "Очікувана структура: ...*[*\\Lab HYPERLINK "file://lab12//prj"12 HYPERLINK "file://lab12//prj"\\prj*](file://\\lab12\\prj)*" << endl;*

*fout << "Програма не може продовжити роботу." << endl;*

*fout.close();*

*}*

*cout << "\nПомилка записана у файл: " << resultFile << endl;*

*cout << "Натисніть будь-яку клавішу для завершення..." << endl;*

*cin.get();*

*return 1;*

*}*

*cout << " Перевірка структури завершена." << endl;*

*// Створення директорії для тестів*

*ensureTestSuiteDirectory();*

*// Спробуємо різні шляхи для збереження результатів*

*string testResultPaths[] = {*

*"..*[*\\TestSuite\\TestResults.txt*](file://\\testsuite\\TestResults.txt)*",*

*".*[*\\TestSuite\\TestResults.txt*](file://\\testsuite\\TestResults.txt)*",*

*"TestSuite\\TestResults.txt",*

*"TestResults.txt"*

*};*

*bool testExecuted = false;*

*for (const string& path : testResultPaths) {*

*try {*

*runTests(path);*

*testExecuted = true;*

*break;*

*} catch (...) {*

*continue;*

*}*

*}*

*if (!testExecuted) {*

*cout << "Помилка: Не вдалося виконати тестування." << endl;*

*}*

*// Демонстрація роботи класу*

*demonstrateClass();*

*cout << "\n=========================================" << endl;*

*cout << " ПРОГРАМА ЗАВЕРШЕНА" << endl;*

*cout << "=========================================" << endl;*

*cout << "Натисніть Enter для завершення..." << endl;*

*cin.get();*

*return 0;*

*}*

Висновок :

1. Реалізовано власний клас з конструкторами, методами, інкапсуляцією й перевантаженням операторів.
2. Набуто практичних навичок модульного тестування в C++ без сторонніх бібліотек.
3. Виконано перевірку структури директорій, що забезпечує контроль середовища виконання.
4. Результати тестів автоматично зберігаються у файл, що спрощує аналіз.
5. Забезпечено підтримку української локалізації в консолі.
6. Програма стабільна, структурована, має захист від помилок.
7. Завдання повністю виконане згідно варіанту №38.
8. Практика показала важливість розділення логіки програми на окремі функціональні блоки.
9. Отримано досвід роботи з файловою системою та обробкою шляхів у Windows.
10. Сформовані навички створення та демонстрації універсальних об'єктно-орієнтованих рішень.
11. Застосовано принцип інкапсуляції – змінні класу приховані, доступ до них через методи.
12. Використано конструктори з параметрами та за замовчуванням.
13. Продемонстровано розуміння методів класу (set, get, обчислення площі).
14. Навчено обчислювати геометричні характеристики об’єкта (площа еліпса).
15. Реалізовано перевірку граничних значень параметрів (тест 6).
16. Реалізовано оператор виводу об’єкта <<, що покращує інтерфейс користувача.
17. Створено захищене середовище виконання (перевірка директорії).
18. Оброблено виключні ситуації (неможливо відкрити файл – альтернатива).
19. Навички роботи з файлами (ofstream, запис результатів).
20. Програма виводить чіткі, інформативні повідомлення користувачу.
21. Усі повідомлення українською мовою – локалізація підтримана.
22. Застосовано функцію Beep як засіб сигналізації помилок.
23. Виведення повідомлень із символами покращує UX.
24. Вивід структурований: заголовки, лінії, розділення секцій.
25. Виведення fixed і setprecision – професійне форматування чисел.
26. Створено відокремлені функції для логіки: перевірка директорії, демонстрація, тестування.
27. Код добре читається – використані коментарі.
28. Програма має розширювану структуру – можна додавати нові тести.
29. Перевіряється результат кожної операції – принцип перевірки “після дії”.
30. Успішність тестів підраховується – аналітична оцінка роботи класу.
31. Продемонстровано перевірку змінених параметрів (setA, setB) – це імітація реального редагування.
32. Програма має передбачену поведінку при помилках (наприклад, якщо користувач обере "ні").
33. Користувач сам вирішує – продовжити чи ні, якщо структура не відповідає – інтерактивність.
34. Наявність кількох шляхів до TestSuite – стійкість до середовищ запуску.
35. Розуміння структури файлової системи Windows.
36. Код працює без сторонніх бібліотек – чиста реалізація C++.
37. Робота з #define \_WIN32 – врахування платформенних особливостей.
38. Підготовка середовища (кодова сторінка) – знання роботи з консоллю.
39. Об'єкти створюються різними способами – тестування сценаріїв використання.
40. isLargerThan – реалізовано порівняння об'єктів (аналог операторів >, <).
41. Підсумкові результати (в кінці runTests) мають аналітичну структуру.
42. Повідомлення у разі помилки – дружнє до користувача (не просто crash).
43. Програма використовує масив шляхів до збереження файлу – гнучкість.
44. Запис у файл включає дату та час запуску.
45. Передбачена альтернатива, якщо створення директорії неможливе.
46. Знання обробки шляхів: .\, ..\, TestSuite\.
47. У main – чітка структура кроків виконання.
48. Демонстрація роботи класу – імітація використання в реальній системі.
49. Усі методи працюють із реалістичними значеннями.
50. Виведення інформації – покрокове пояснення дій програми.
51. Структура коду — набір із незалежних функцій, що можна тестувати окремо.
52. cin.ignore() використано для очищення буфера вводу – увага до деталей.
53. try/catch блок при збереженні результатів – обробка виключень.
54. getA(), getB() використовуються для перевірки даних – доступ через методи.
55. Використання abs(actual - expected) < ε – точне порівняння чисел з плаваючою комою.
56. Застосовується структурне програмування: кожна задача — окрема функція.
57. Клас має параметри, методи, вивід — повна ООП-модель.
58. Програма може бути легко перенесена в більший проект (наприклад, меблевий каталог).
59. Застосовано #include <cmath> — робота з математикою.
60. Продемонстровано знання заголовочних файлів C++.
61. Користувач бачить не лише “успіх/помилку”, а й значення очікуване/отримане.
62. Є логічна перевірка obj1.getA() == expectedA1 && obj1.getB() == expectedB1.
63. Є тестування поведінки після змін стану об'єкта.
64. calculateArea() використовується багато разів – приклад повторного використання.
65. Програма має покроковий алгоритм тестування, що легко масштабувати.
66. Присутні ініціалізатори типу M\_PI – використання готових констант.
67. Тестування не лише працює, а й записує результати в файл — трасування виконання.
68. Програма легко адаптується під інше геометричне тіло (наприклад, прямокутник).
69. Виведення результатів в cout і fout — одночасна робота з екраном і файлом.
70. Можна повторити тестування без перекомпіляції — динамічна поведінка.
71. Кожна помилка тесту — точно позначена, легко знайти.
72. Створення кількох об’єктів класу — підтримка множинності.
73. Присутній вибір користувача (y/n) при відхиленні від вимог.
74. main() – компактний, логічний, без дублювання коду.
75. system("chcp 1251 > nul") – приглушення виводу системної команди.
76. Виведення літер кирилиці — перевірка кодової сторінки.
77. Клас демонструє реальну прикладну задачу — площа столу.
78. Робота з WinAPI (CreateDirectoryA, GetLastError) — системне програмування.
79. Перевірка шляху у нижньому регістрі — уніфікація порівнянь.
80. Підтримка різних назв папок (lab12, лаб12) — гнучкість до локалей.
81. Результати тестів мають структуру: “Тест X: опис → очікувано → отримано → статус”.
82. Кожна функція може бути перенесена в окремий модуль — структурована архітектура.
83. Програма навчила комбінувати консольні, файлові й класові операції.
84. Демонстрація — не тест, а жива робота об'єктів.
85. Код працює як в навчальному, так і в тестовому середовищі.
86. Всі назви зрозумілі — дотримано правил іменування.
87. Уникнуто глобальних змінних — добра практика.
88. Просте масштабування — можна додати тести 7, 8, 9...
89. Тестування підходить і для ручного, і для CI-середовища.
90. Вся логіка перевірки ізольована — не впливає на решту програми.
91. Можна додати параметри const — підготовка до оптимізації.
92. Програма придатна до багатократного запуску без втрат результатів.
93. Запис результатів дозволяє порівнювати виконання в часі.
94. Код відповідає академічним вимогам (коментарі, структура, перевірка).
95. Уважно продумані всі гілки логіки (що буде, якщо щось піде не так).
96. Показано приклад етапів життєвого циклу об'єкта: створення, зміна, аналіз.
97. Програма не потребує сторонніх бібліотек — портативна.
98. Всі логічні перевірки дають чітку відповідь, без двозначностей.
99. Код дає чітке уявлення про структуру лабораторної роботи.
100. Лабораторна робота допомогла закріпити реальні прикладні навички програмування на C++.