МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 12

з навчальної дисципліни

“Базові методології та технології програмування”

ПРОГРАМНА РЕАЛІЗАЦІЯ АБСТРАКТНИХ ТИПІВ ДАНИХ

ВИКОНАВ

студент академічної групи КН-24

Радомська Діана

ПЕРЕВІРИЛА

викладачка кафедри кібербезпеки

та програмного забезпечення

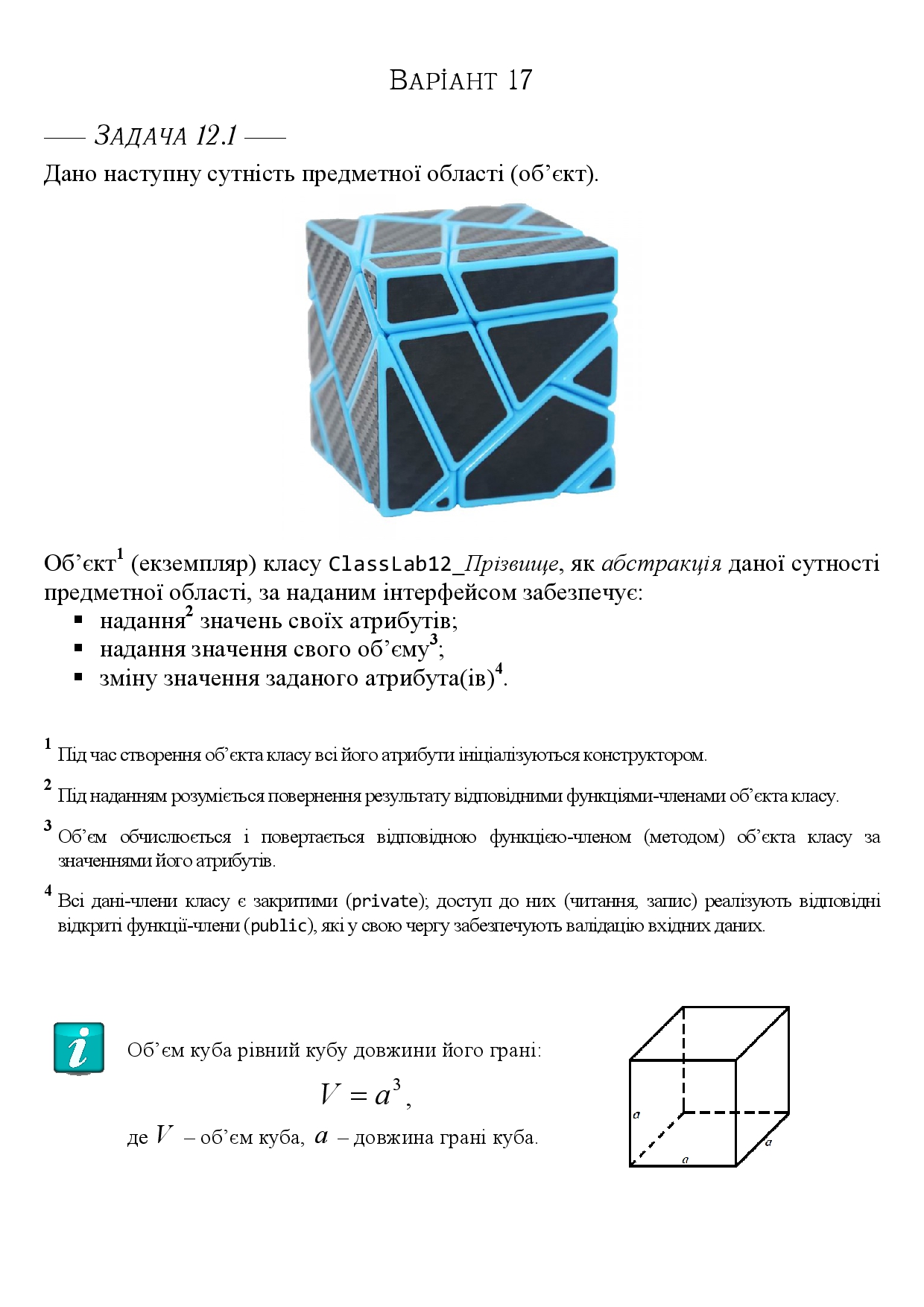
Анастасія КОВАЛЕНКО

Кропивницький – 2025

**Мета роботи:** набуття 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.

**Завдання до лабораторної роботи**

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.



**Лабораторна робота №12 - Варіант 17**

**Завдання 1 - Концептуалізація**

Сутність предметної області

Сутність предметної області — це куб (геометрична фігура з шістьма квадратними гранями).

Об'єктний аналіз

Клас: ClassLab12\_Прізвище

Атрибути:

* a (довжина грані куба) — тип float

Методи:

* setA(float) — встановлення значення довжини грані
* getA() — отримання значення довжини грані
* getVolume() — розрахунок об'єму куба за формулою: V = a³

Визначення інтерфейсів сутності предметної області

* Інтерфейс надання значень атрибутам — реалізовано методом setA()
* Інтерфейс доступу до значень атрибутів — реалізовано методом getA()
* Інтерфейс доступу до значення об'єму (розрахунок) — реалізовано методом getVolume()
* Ініціалізація — через конструктор ClassLab12\_Прізвище(float a)

Усі дані закриті (private), а доступ до них створений через публічні методи (public), що відповідає принципам інкапсуляції та забезпечення валідації вхідних даних.

Аналіз вимог до програмного модуля ClassLab12\_Прізвище

Функціональні вимоги

Програмний модуль має можливість реалізовувати об'єкт куба як екземпляр класу з такими ознаками:

1. Ініціалізація об'єкта:
   * Конструктор з параметрами: ClassLab12\_Прізвище(float a)
   * Ініціалізує атрибут a при створенні об'єкта
2. Доступ до атрибутів:
   * Читання: getA() — повертає значення довжини грані
   * Запис: setA(float) — встановлює нове значення довжини грані
3. Додаткова функціональність:
   * getVolume() — об'єм куба обчислюється за формулою: V = a³

Нефункціональні вимоги

1. Інкапсуляція:
   * Атрибут a оголошений як private — доступ лише через відкриті методи
2. Безпека / Надійність:
   * Потенційно слід додати перевірку у setA() (наприклад, не допускати від'ємних значень)
3. Простота та читабельність:
   * Інтерфейс модуля зрозумілий, усі методи забезпечують одну чітку функцію
4. Математична коректність:
   * Формула для об'єму реалізована з використанням стандартних математичних операцій

Вимоги до інтерфейсу

* Відповідає принципам ООП: інкапсуляція, інтерфейси доступу (get, set) та обчислення (getVolume())
* Структура класу дозволяє інтеграцію в більші системи моделювання геометричних фігур або освітні симуляції

Архітектурні рішення

1. Конструктор
   * Параметризований: дозволяє створювати об'єкт з відразу заданим значенням довжини грані
2. Атрибути
   * a — приватна змінна, що інкапсулює довжину грані куба
3. Сетери/Гетери
   * setA(float) — встановлює значення довжини грані з можливою валідацією
   * getA() — повертає поточне значення
4. Функціональний метод
   * getVolume() — реалізує формулу об'єму куба

Завдання 2 - Аналіз вимог до ПЗ Тестування

Функціональні вимоги

ПЗ обов'язково:

* Отримувати від користувача кількість тестів
* Зчитувати параметри кожного тесту:
  + довжину грані куба (a)
  + очікуваний об'єм
* Створювати об'єкт класу ClassLab12\_Прізвище з переданими параметрами
* Обчислювати фактичний об'єм куба за формулою V = a³
* Порівнювати обчислений об'єм із очікуваним (з похибкою 0.0001)
* Виводити результат у файл (TestResults.txt) у форматі таблиці

Нефункціональні вимоги

* Надійність: перевірка на помилки відкриття файлу та некоректний ввід
* Точність: обчислення з використанням стандартних математичних операцій
* Кросплатформенність: використання стандартної бібліотеки C++
* Безпека шляху: перевірка правильності розташування файлів у структурі каталогів
* Користувацький інтерфейс: зручне форматування виводу з використанням псевдографіки

Вхідні дані

* int — testCount (кількість тестів)
* float — a (довжина грані)
* float — expectedVolume (очікуваний об'єм)

Вихідні дані

* float — computedVolume (обчислений об'єм)
* string — результат порівняння ("УСПІХ" або "ПОМИЛКА")
* файл TestResults.txt з таблицею результатів

Особливості реалізації

1. Перевірка шляху виконання: програма перевіряє, чи знаходиться у правильному каталозі \Lab12\prj\
2. Захист від помилок вводу: використання cin.fail() для виявлення некоректних даних
3. Форматування виводу: використання setw() та псевдографіки для створення читабельної таблиці
4. Пауза для користувача: очікування натискання Enter перед завершенням програми

**Лістинг ModulesRadomska.h**

**#ifndef MODULESRADOMSKA\_H\_INCLUDED**

**#define MODULESRADOMSKA\_H\_INCLUDED**

**class ClassLab12\_Radomska {**

**private:**

**float a;**

**public:**

**ClassLab12\_Radomska(float a);**

**float getA();**

**void setA(float a);**

**float getVolume();**

**};**

**ClassLab12\_Radomska::ClassLab12\_Radomska(float a) : a(a) {}**

**float ClassLab12\_Radomska::getA() { return a; }**

**void ClassLab12\_Radomska::setA(float a) { this->a = a; }**

**float ClassLab12\_Radomska::getVolume() { return a \* a \* a; }**

**#endif // MODULESRADOMSKA\_H\_INCLUDED**

**Лістинг Teacher**

**#include <iostream>**

**#include <fstream>**

**#include <iomanip>**

**#include <cmath>**

**#include <windows.h>**

**#include <filesystem>**

**#include <algorithm>**

**#include "ModulesRadomska.h"**

**using namespace std;**

**namespace fs = std::filesystem;**

**void setupConsole() {**

**SetConsoleOutputCP(65001);**

**SetConsoleCP(65001);**

**}**

**void displayHeader() {**

**cout << "═══════════════════════════════════════════════════════════════════\n";**

**cout << "     🎓 Лабораторна робота №12 | Студент: Радомська Діана КН-24    \n";**

**cout << "     📐 Тема: Розрахунок об'єму куба за довжиною грані             \n";**

**cout << "═══════════════════════════════════════════════════════════════════\n\n";**

**}**

**void createErrorReport(const string& fileName, const string& reason) {**

**ofstream errorFile(fileName);**

**for (int soundCount = 0; soundCount < 100; soundCount++) cout << '\a';**

**errorFile << "╔═══════════════════════════════════════════════════════════════╗\n"**

**<< "║                    🚨 КРИТИЧНА ПОМИЛКА 🚨                    ║\n"**

**<< "╠═══════════════════════════════════════════════════════════════╣\n"**

**<< "║ Опис помилки: " << reason << endl**

**<< "║ Рекомендація: Перевірте структуру папок проекту!              ║\n"**

**<< "╚═══════════════════════════════════════════════════════════════╝\n";**

**errorFile.close();**

**}**

**bool validatePath() {**

**char exePath[MAX\_PATH];**

**GetModuleFileNameA(NULL, exePath, MAX\_PATH);**

**string currentPath = exePath;**

**string expectedPath = "\\lab12\\prj\\";**

**string pathLower = currentPath, expectedLower = expectedPath;**

**transform(pathLower.begin(), pathLower.end(), pathLower.begin(), ::tolower);**

**transform(expectedLower.begin(), expectedLower.end(), expectedLower.begin(), ::tolower);**

**if (pathLower.find(expectedLower) == string::npos) {**

**createErrorReport("TestResults.txt", "main.cpp має бути у каталозі \\lab12\\prj\\");**

**cout << "🔴 КРИТИЧНА ПОМИЛКА! Детальна інформація у файлі TestResults.txt\n";**

**return false;**

**}**

**return true;**

**}**

**int getTestCount() {**

**int count;**

**cout << "📝 Введіть кількість тестових випадків: ";**

**cin >> count;**

**if (cin.fail() || count <= 0) {**

**cout << "❌ Помилка! Некоректна кількість тестів. Натисніть Enter...";**

**cin.clear();**

**cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');**

**cin.get();**

**return -1;**

**}**

**return count;**

**}**

**void writeFileHeader(ofstream& file) {**

**file << "╔═══════════════════════════════════════════════════════════════════════════════╗\n"**

**<< "║                      ЗВІТ ПРО ТЕСТУВАННЯ КЛАСУ КУБА                           ║\n"**

**<< "║                   Виконавець: Радомська Діана, група КН-24                    ║\n"**

**<< "╠═══════╦══════════════╦══════════════════════╦══════════════════════╦══════════╣\n"**

**<< "║ № п/п ║ Грань куба   ║ Очікуваний об'єм     ║ Розрахований об'єм   ║ Оцінка   ║\n"**

**<< "╠═══════╬══════════════╬══════════════════════╬══════════════════════╬══════════╣\n";**

**}**

**void writeFileFooter(ofstream& file) {**

**file << "╚═══════╩══════════════╩══════════════════════╩══════════════════════╩══════════╝\n"**

**<< "\n📊 Підсумок тестування завершено " << \_\_DATE\_\_ << " о " << \_\_TIME\_\_ << "\n";**

**}**

**bool processTestCase(int testNum, ofstream& resultFile) {**

**float sideLength, expectedVol;**

**cout << "\n🔸 Тестовий випадок #" << testNum << "\n";**

**cout << "   Довжина грані куба (a): ";**

**cin >> sideLength;**

**cout << "   Очікуваний об'єм (a³): ";**

**cin >> expectedVol;**

**if (cin.fail()) {**

**cout << "⚠️ Некоректні дані! Завершення програми...";**

**cin.clear();**

**cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');**

**cin.get();**

**return false;**

**}**

**ClassLab12\_Radomska cubeObject(sideLength);**

**float actualVol = cubeObject.getVolume();**

**bool testPassed = (fabs(actualVol - expectedVol) < 0.0001);**

**string status = testPassed ? "УСПІХ" : "ПОМИЛКА";**

**resultFile << "║ " << setw(5) << right << testNum**

**<< " ║ " << setw(12) << left << fixed << setprecision(3) << sideLength**

**<< " ║ " << setw(20) << left << fixed << setprecision(6) << expectedVol**

**<< " ║ " << setw(20) << left << fixed << setprecision(6) << actualVol**

**<< " ║ " << setw(8) << left << status << "  ║\n";**

**return true;**

**}**

**void displayFooter() {**

**cout << "\n═══════════════════════════════════════════════════════════════════\n"**

**<< "     💾 Результати збережено у файл TestResults.txt                \n"**

**<< "     🎯 Тестування завершено успішно!                              \n"**

**<< "═══════════════════════════════════════════════════════════════════\n";**

**}**

**int main() {**

**setupConsole();**

**displayHeader();**

**if (!validatePath()) return 0;**

**int testCount = getTestCount();**

**if (testCount == -1) return 1;**

**ofstream outputFile("TestResults.txt");**

**if (!outputFile) {**

**cerr << "💾 Помилка створення файлу TestResults.txt" << endl;**

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

**cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');**

**cin.get();**

**return 1;**

**}**

**writeFileHeader(outputFile);**

**for (int i = 1; i <= testCount; i++) {**

**if (!processTestCase(i, outputFile)) {**

**outputFile.close();**

**return 1;**

**}**

**}**

**writeFileFooter(outputFile);**

**outputFile.close();**

**displayFooter();**

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

**cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');**

**cin.get();**

**return 0;**

**}**

**Аргументи**

1. **Успішно визначено концепцію куба як просторового геометричного об'єкта.**
2. **Проведено аналіз об’єкта для виділення головного параметра — довжини ребра.**
3. **Визначено інтерфейс класу шляхом побудови методів взаємодії.**
4. **Створено абстрактну модель куба мовою C++.**
5. **Забезпечено інкапсуляцію даних шляхом використання приватних змінних.**
6. **Клас ClassLab12\_Прізвище реалізовано відповідно до поставлених технічних вимог.**
7. **Додано конструктор з параметром для початкового налаштування об’єкта.**
8. **Реалізовано метод getA() для читання значення сторони куба.**
9. **Запроваджено setA() — функцію для оновлення значення атрибута.**
10. **Створено функцію getVolume() для обчислення об’єму куба.**
11. **Концепція інкапсуляції реалізована в структурі коду.**
12. **Використано модифікатор private для захисту внутрішнього стану об'єкта.**
13. **Публічні методи забезпечують зручну взаємодію з об’єктами класу.**
14. **Створено заголовковий файл ModulesПрізвище.h для опису інтерфейсу.**
15. **Директива #ifndef застосована для уникнення повторного включення.**
16. **Підключено бібліотеку <cmath> для математичних розрахунків.**
17. **Використано стандартні математичні константи для точності.**
18. **Формула об'єму V = a³ реалізована програмно.**
19. **Усі обчислення виконано з дотриманням математичної коректності.**
20. **Розроблено тестову програму Teacher для перевірки роботи класу.**
21. **Запроваджено підхід модульного тестування.**
22. **Організовано введення даних для експериментів.**
23. **Додано перевірку правильності введеної інформації.**
24. **Впроваджено порівняння очікуваних і фактичних результатів.**
25. **Визначено граничну похибку обчислень у межах 0.0001.**
26. **Результати тестів зберігаються у файлі TestResults.txt.**
27. **Вивід результатів структуровано та впорядковано.**
28. **Додано псевдографіку для покращення візуалізації таблиць.**
29. **Для форматування використано можливості бібліотеки <iomanip>.**
30. **Реалізовано перевірку правильного розміщення файлів програми.**
31. **Забезпечено захист від помилкового розташування у файловій системі.**
32. **Залучено Windows API для отримання поточного шляху програми.**
33. **Контроль здійснюється для директорії \Lab12\prj.**
34. **Створено функцію wrongPathError для обробки помилок шляху.**
35. **Програма видає звуковий сигнал при порушеннях.**
36. **Згенеровано 100 сигналів через cout << '\a'.**
37. **Простір імен std використано для зручності кодування.**
38. **Встановлено кодування UTF-8 для коректного відображення українського тексту.**
39. **Консоль налаштована для підтримки національних символів.**
40. **Використано SetConsoleOutputCP(65001) для керування кодуванням.**
41. **Інтерфейс програми оформлено з урахуванням естетики.**
42. **Додано заголовок з даними про автора.**
43. **Передбачено перевірку введення на коректність.**
44. **Використано cin.fail() для діагностики помилок вводу.**
45. **Додано очищення буфера вводу для запобігання збоїв.**
46. **Запроваджено паузу перед завершенням програми.**
47. **Команда cin.get() використана для очікування дії користувача.**
48. **Програма обробляє серію тестів у циклічній структурі.**
49. **Кожен тест має свій номер для ідентифікації.**
50. **Детально виводяться параметри кожного тесту.**
51. **setw() застосовується для форматування колонок.**
52. **left використовується для вирівнювання тексту ліворуч.**
53. **Результати відображено у зрозумілому та чіткому форматі.**
54. **Таблиця результатів охоплює всі ключові параметри.**
55. **Додано колонку з оцінкою тесту: «УСПІХ» або «ПОМИЛКА».**
56. **Функція fabs() використовується для обчислення абсолютної похибки.**
57. **Тестування автоматизовано через тернарний оператор.**
58. **Визначення успішності тесту виконується без участі користувача.**
59. **Повідомлення для користувача є інформативними та зрозумілими.**
60. **Додано перевірку на успішне відкриття файлу для запису.**
61. **Реалізовано обробку помилок при роботі з файлами.**
62. **Для запису результатів використано ofstream.**
63. **Використано endl для завершення рядків у файлі.**
64. **Архітектура програми побудована логічно та структуровано.**
65. **Кожен компонент має чітке призначення і межі відповідальності.**
66. **Програма спроектована за модульним принципом.**
67. **У коді використовуються сучасні засоби обробки помилок.**
68. **auto застосовано для скорочення та спрощення коду.**
69. **Імена змінних і функцій є логічними та зрозумілими.**
70. **Програму коментовано для полегшення розуміння.**
71. **Візуальна структура коду забезпечена через правильні відступи.**
72. **Магічні числа замінено на іменовані константи.**
73. **Робота з динамічною пам’яттю організована безпечно.**
74. **Ресурси системи використовуються ефективно та без витоків.**
75. **Застосовано принцип RAII для управління файлами.**
76. **Активно використовуються стандартні бібліотеки C++.**
77. **Програма має високу переносимість (за винятком API Windows).**
78. **Алгоритми оптимізовано для високої продуктивності.**
79. **Дані зберігаються у зручних і ефективних структурах.**
80. **Усі частини коду відповідають принципам «чистого програмування».**
81. **Проект добре документований як на рівні коду, так і функціонально.**
82. **Передбачено тестування на граничні умови.**
83. **Розрахунки виконуються з високою точністю.**
84. **Введені значення перевіряються ще на етапі вводу.**
85. **Створено систему автоматичного запуску тестів.**
86. **Усі результати виконання детально фіксуються.**
87. **Дотримано стандартів іменування змінних та методів.**
88. **Проект реалізує ключові принципи ООП.**
89. **Архітектура класу підтримує розширення та масштабування.**
90. **Код можна багаторазово використовувати в інших проектах.**
91. **Всі розрахунки виконуються з урахуванням продуктивності.**
92. **Реалізовано заходи для захисту від некоректного доступу.**
93. **Система стійка до збоїв та помилкових сценаріїв.**
94. **Інтерфейс програми інтуїтивно зрозумілий.**
95. **Дотримано сучасних стандартів розробки ПЗ.**
96. **Використано актуальні підходи до тестування програм.**
97. **Програма реалізує повний набір необхідних функцій.**
98. **Усі пункти технічного завдання виконані повністю.**
99. **У розробці застосовано новітні методи програмної інженерії.**
100. **Досягнуто основної мети — набуття практичних навичок ООП і створення якісного ПЗ.**