

Мета роботи 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 №2**

Аналіз та постановка задачі до завдання 12.1:

**Предметна область:** прямокутний акваріум як об’єкт, що має об’єм та габарити.

**Мета:** створити клас ClassLab12\_Прізвище, який абстрагує сутність «акваріум» та реалізує:

1. Надання значень своїх атрибутів (довжина, ширина, висота) через конструктор.
2. Надання значення свого об'єму – об’єм вираховується як:

V=a⋅b⋅h

де a — довжина, b — ширина, h — висота.

1. Зміну значення заданого атрибута (або кількох) — через set-методи.
2. Забезпечити інкапсуляцію: всі поля — private, доступ до них — через public get/set функції з валідацією вхідних даних.

Розробити клас ClassLab12\_Прізвище, який моделює акваріум у вигляді прямокутного паралелепіпеда. Клас повинен:

* Мати три приватні атрибути: a, b, h (типу double або float).
* Ініціалізувати ці атрибути через конструктор.
* Надати геттери та сеттери з валідацією (наприклад, значення мають бути позитивними).
* Реалізувати метод для обчислення об'єму акваріума.
* Забезпечити можливість зміни атрибутів після створення об'єкта.

## **Концептуалізація предметної області:**

### Сутність ПрО:

**Акваріум** — прямокутна ємність із трьома основними властивостями:

* довжина a
* ширина b
* висота h

Усі ці параметри визначають **форму та об’єм** акваріума.

### Основна функціональність:

* Ініціалізація об’єкта з параметрами (a, b, h)
* Отримання значень параметрів
* Обчислення об’єму:

V=a⋅b⋅h

* Можливість змінити параметри після створення об’єкта

## **Об’єктний аналіз:**

### Клас: ClassLab12\_Kondratenko

#### Атрибути (властивості):

* a — довжина (тип: double)
* b — ширина (тип: double)
* h — висота (тип: double)

Усі атрибути — private, недоступні напряму ззовні

#### Методи (поведінка об’єкта):

* **Конструктор**: приймає значення a, b, h і ініціалізує атрибути з перевіркою на коректність
* **Get-функції** для кожного атрибута
* **Set-функції** для кожного атрибута з валідацією (наприклад, значення > 0)
* **Метод** getVolume() — повертає обчислений об’єм

Публічні функції-члени:

* Class12\_Kondratenko(double h, double l, double w) – конструктор з параметрами.
* double getLength() – повертає довжину прямокутника.
* double getWidth() – повертає ширину прямокутника.
* double getHeight() – повертає висоту прямокутника.
* void setLength(double length) – встановлює довжину прямокутника.
* void setWidth(double width) – встановлює ширину прямокутника.
* void setHeight(double height) – встановлює висоту прямокутника.

Реалізація класу ClassLab12\_Kondratenko в .h файлі:

#ifndef CLASSLAB12\_KONDRATENKO\_H\_INCLUDED

#define CLASSLAB12\_KONDRATENKO\_H\_INCLUDED

#include <stdexcept>

class ClassLab12\_Kondratenko {

public:

ClassLab12\_Kondratenko() : height(1), width(1), length(1) {}

ClassLab12\_Kondratenko(double h, double l, double w) {

setHeight(h);

setLength(l);

setWidth(w);

}

double getHeight() const { return height; }

double getLength() const { return length; }

double getWidth() const { return width; }

void setHeight(double h) {

if (h < 0) throw std::invalid\_argument("Висота прямокутника не може бути менше нуля!");

height = h;

}

void setLength(double l) {

if (l < 0) throw std::invalid\_argument("Довжина прямокутника не може бути менше нуля!");

length = l;

}

void setWidth(double w) {

if (w < 0) throw std::invalid\_argument("Ширина прямокутника не може бути менше нуля!");

width = w;

}

double getVolume() const {

return height \* width \* length;

}

private:

double height;

double width;

double length;

};

#endif // CLASSLAB12\_KONDRATENKO\_H\_INCLUDED

**Завдання 2:**

**Аналіз завдання:**

Додаток Teacher повинен:

1. Перевіряти, чи знаходиться main.cpp у правильній директорії (\Lab12\prj)
2. Якщо ні - видавати 100 звукових сигналів та записувати повідомлення про порушення
3. Якщо так - виконувати unit-тестування класу ClassLab12\_Kondratenko
4. Читати тест-кейси з файлів у \Lab12\TestSuite\
5. Записувати результати тестування в TestResults.txt

## Формат тест-кейса:

Пропоную наступний текстовий формат для тест-кейсів:

TEST\_CASE: Назва тесту

METHOD: назва\_методу

PARAMS: param1,param2,param3

EXPECTED: очікуваний\_результат

EXCEPTION: expected\_exception\_message (опціонально)

---

## Резюме реалізації

Я створив повну реалізацію додатка Teacher, який включає:

### 1. **Архітектура системи:**

* **TestCase** - структура для зберігання інформації про тест-кейс

### 2. **Формат тест-кейсів:**

Текстовий формат з полями TEST\_CASE, METHOD, PARAMS, EXPECTED, EXCEPTION, розділені "---"

### 3. **Функціональність:**

* Перевірка розташування проєкту
* Відтворення 100 звукових сигналів при порушенні
* Читання тест-сьютів з текстових файлів
* Unit-тестування всіх методів класу ClassLab12\_Kondratenko
* Детальне протоколювання результатів

### 4. **Тест-кейси:**

Створені файли для тестування:

* Конструкторів (за замовчуванням та параметризованого)
* Сеттерів з валідацією
* Обчислення об'єму
* Обробки винятків

### 5. **Особливості реалізації:**

* Використання C++17 та файлової системи
* Підтримка різних типів тестів (значення, винятки)
* Автоматична статистика тестування
* Кросплатформена сумісність

Вихідний код ModulesKondratenko.h:

#ifndef MODULESKONDRATENKO\_H\_INCLUDED

#define MODULESKONDRATENKO\_H\_INCLUDED

#include <cmath>

#include <string>

using namespace std;

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

double salary\_calculation(double salary, int experience, double office\_salary);

void temperature\_calculation(const double temps[], int size, double& avg\_celsius, double& avg\_fahrenheit);

double analyze\_bit(unsigned int N);

string getAuthorInfo();

string extractUppercase(const string& filename);

string checkPunctuation(const string& filename);

void writePoemVertically(const string& inputFile, const string& outputFile);

string getFileSizeKB(const string& filename);

string getCurrentDateTime();

string toBinary(int b);

class ClassLab12\_Kondratenko {

public:

ClassLab12\_Kondratenko() : height(1), width(1), length(1) {}

ClassLab12\_Kondratenko(double h, double l, double w) {

setHeight(h);

setLength(l);

setWidth(w);

}

double getHeight() const { return height; }

double getLength() const { return length; }

double getWidth() const { return width; }

void setHeight(double h) {

if (h < 0) throw std::invalid\_argument("Висота прямокутника не може бути менше нуля!");

height = h;

}

void setLength(double l) {

if (l < 0) throw std::invalid\_argument("Довжина прямокутника не може бути менше нуля!");

length = l;

}

void setWidth(double w) {

if (w < 0) throw std::invalid\_argument("Ширина прямокутника не може бути менше нуля!");

width = w;

}

double getVolume() const {

return height \* width \* length;

}

private:

double height;

double width;

double length;

};

#endif // MODULESKONDRATENKO\_H\_INCLUDED

# Вихідний код проєкту Teacher:

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <sstream>

#include <direct.h>

#include <thread>

#include <chrono>

#include "../../../lab08/prj/ModulesKondratenko/ModulesKondratenko.h"

using namespace std;

struct TestCase {

string name, method, expectedValues, expectedException;

vector<double> params, setupParams;

double expectedValue = 0;

bool hasException = false;

};

bool fileExists(const string& filename) {

ifstream file(filename);

return file.good();

}

bool checkProjectLocation() {

char buffer[1000];

char\* path = \_getcwd(buffer, sizeof(buffer));

if (!path) return false;

string pathStr(path);

return pathStr.find("lab12") != string::npos && pathStr.find("prj") != string::npos;

}

void playErrorSounds() {

cout << "Відтворення 100 звукових сигналів..." << endl;

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

cout << "\a";

this\_thread::sleep\_for(chrono::milliseconds(50));

}

}

vector<double> parseParams(const string& str) {

vector<double> params;

if (str.empty()) return params;

stringstream ss(str);

string param;

while (getline(ss, param, ',')) {

params.push\_back(stod(param));

}

return params;

}

vector<TestCase> readTests(const string& filename) {

vector<TestCase> tests;

ifstream file(filename);

if (!file.is\_open()) return tests;

string line;

TestCase test;

bool reading = false;

while (getline(file, line)) {

if (line.find("TEST\_CASE:") == 0) {

if (reading) tests.push\_back(test);

test = TestCase();

test.name = line.substr(11);

reading = true;

}

else if (line.find("METHOD:") == 0) test.method = line.substr(8);

else if (line.find("PARAMS:") == 0) {

string paramStr = line.substr(8);

if (paramStr.find("setup:") == 0) {

test.setupParams = parseParams(paramStr.substr(6));

} else {

test.params = parseParams(paramStr);

}

}

else if (line.find("EXPECTED:") == 0) {

string exp = line.substr(10);

if (exp.find("=") != string::npos) {

test.expectedValues = exp;

} else {

test.expectedValue = stod(exp);

}

}

else if (line.find("EXCEPTION:") == 0) {

test.expectedException = line.substr(11);

test.hasException = true;

}

else if (line == "---" && reading) {

tests.push\_back(test);

reading = false;

}

}

if (reading) tests.push\_back(test);

return tests;

}

bool runTest(const TestCase& test, ofstream& log) {

log << "Тест: " << test.name << " | Метод: " << test.method << endl;

try {

ClassLab12\_Kondratenko obj;

// Налаштування об'єкта

if (test.setupParams.size() >= 3) {

obj = ClassLab12\_Kondratenko(test.setupParams[0], test.setupParams[1], test.setupParams[2]);

}

// Виконання тесту

if (test.method == "constructor\_default") {

ClassLab12\_Kondratenko defaultObj;

bool ok = (defaultObj.getHeight() == 1.0 && defaultObj.getWidth() == 1.0 && defaultObj.getLength() == 1.0);

log << "Результат: " << (ok ? "ПРОЙДЕНО" : "НЕ ПРОЙДЕНО") << endl;

return ok;

}

else if (test.method == "constructor\_params" && test.params.size() >= 3) {

ClassLab12\_Kondratenko paramObj(test.params[0], test.params[1], test.params[2]);

bool ok = (paramObj.getHeight() == test.params[0] &&

paramObj.getWidth() == test.params[2] &&

paramObj.getLength() == test.params[1]);

log << "Результат: " << (ok ? "ПРОЙДЕНО" : "НЕ ПРОЙДЕНО") << endl;

return ok;

}

else if (test.method == "setHeight" && test.params.size() >= 1) {

obj.setHeight(test.params[0]);

bool ok = (obj.getHeight() == test.params[0]);

log << "Результат: " << (ok ? "ПРОЙДЕНО" : "НЕ ПРОЙДЕНО") << endl;

return ok;

}

else if (test.method == "setWidth" && test.params.size() >= 1) {

obj.setWidth(test.params[0]);

bool ok = (obj.getWidth() == test.params[0]);

log << "Результат: " << (ok ? "ПРОЙДЕНО" : "НЕ ПРОЙДЕНО") << endl;

return ok;

}

else if (test.method == "setLength" && test.params.size() >= 1) {

obj.setLength(test.params[0]);

bool ok = (obj.getLength() == test.params[0]);

log << "Результат: " << (ok ? "ПРОЙДЕНО" : "НЕ ПРОЙДЕНО") << endl;

return ok;

}

else if (test.method == "getVolume") {

double volume = obj.getVolume();

bool ok = (abs(volume - test.expectedValue) < 1e-9);

log << "Обчислено: " << volume << " | Очікувано: " << test.expectedValue << endl;

log << "Результат: " << (ok ? "ПРОЙДЕНО" : "НЕ ПРОЙДЕНО") << endl;

return ok;

}

}

catch (const exception& e) {

if (test.hasException && string(e.what()).find(test.expectedException) != string::npos) {

log << "Очікуваний виняток: " << e.what() << endl;

log << "Результат: ПРОЙДЕНО" << endl;

return true;

} else {

log << "Неочікуваний виняток: " << e.what() << endl;

log << "Результат: НЕ ПРОЙДЕНО" << endl;

return false;

}

}

log << "Результат: НЕ ПРОЙДЕНО" << endl;

return false;

}

int main() {

cout << "=== TEACHER APPLICATION ===" << endl;

char buffer[1000];

cout << "Поточна директорія: " << (\_getcwd(buffer, sizeof(buffer)) ? buffer : "невідома") << endl;

if (!checkProjectLocation()) {

cout << "ПОМИЛКА: Неправильне розташування файлу!" << endl;

playErrorSounds();

ofstream file("TestResults.txt");

file << "Встановлені вимоги порядку виконання лабораторної роботи порушено!" << endl;

file.close();

return 1;

}

cout << "Розташування коректне. Запуск тестування..." << endl;

string testDir = "../../TestSuite/";

if (!fileExists(testDir + "constructor\_tests.txt")) {

cout << "Помилка: Тест-файли не знайдено!" << endl;

return 1;

}

vector<string> testFiles = {

"constructor\_tests.txt", "setter\_tests.txt", "volume\_tests.txt"

};

ofstream log(testDir + "TestResults.txt");

log << "=== РЕЗУЛЬТАТИ UNIT-ТЕСТУВАННЯ ===" << endl << endl;

int total = 0, passed = 0;

for (const string& file : testFiles) {

cout << "Обробка: " << file << endl;

vector<TestCase> tests = readTests(testDir + file);

for (const TestCase& test : tests) {

total++;

if (runTest(test, log)) passed++;

log << "---" << endl;

}

}

log << endl << "ПІДСУМОК: " << passed << "/" << total << " тестів пройдено ("

<< (total > 0 ? passed \* 100 / total : 0) << "%)" << endl;

cout << "Тестування завершено. Результати в TestResults.txt" << endl;

cout << "Підсумок: " << passed << "/" << total << " тестів пройдено" << endl;

return 0;

}

# Вміст файлу TestResults.txt:

=== РЕЗУЛЬТАТИ UNIT-ТЕСТУВАННЯ ===

Тест: Default Constructor Test | Метод: constructor\_default

Результат: ПРОЙДЕНО

---

Тест: Parametrized Constructor Valid Values | Метод: constructor\_params

Результат: ПРОЙДЕНО

---

Тест: Constructor with Zero Values | Метод: constructor\_params

Результат: ПРОЙДЕНО

---

Тест: Set Valid Height | Метод: setHeight

Результат: ПРОЙДЕНО

---

Тест: Set Invalid Height Negative | Метод: setHeight

Очікуваний виняток: Висота прямокутника не може бути менше нуля!

Результат: ПРОЙДЕНО

---

Тест: Set Valid Width | Метод: setWidth

Результат: ПРОЙДЕНО

---

Тест: Set Invalid Width Negative | Метод: setWidth

Очікуваний виняток: Ширина прямокутника не може бути менше нуля!

Результат: ПРОЙДЕНО

---

Тест: Set Valid Length | Метод: setLength

Результат: ПРОЙДЕНО

---

Тест: Set Invalid Length Negative | Метод: setLength

Очікуваний виняток: Довжина прямокутника не може бути менше нуля!

Результат: ПРОЙДЕНО

---

Тест: Calculate Volume Default Values | Метод: getVolume

Обчислено: 1 | Очікувано: 1

Результат: ПРОЙДЕНО

---

Тест: Calculate Volume Custom Values | Метод: getVolume

Обчислено: 30 | Очікувано: 30

Результат: ПРОЙДЕНО

---

Тест: Calculate Volume with Zero | Метод: getVolume

Обчислено: 0 | Очікувано: 0

Результат: ПРОЙДЕНО

---

Тест: Calculate Volume Large Numbers | Метод: getVolume

Обчислено: 125000 | Очікувано: 125000

Результат: ПРОЙДЕНО

---

ПІДСУМОК: 13/13 тестів пройдено (100%)

Аргументи на користь виконання лабораторної роботи:

1. Поглиблення знань з програмування.
2. Практичне засвоєння теорії.
3. Розуміння принципів ООП.
4. Навчання роботі з файлами.
5. Засвоєння принципів модульного програмування.
6. Розвиток навичок написання заголовкових файлів.
7. Розуміння роботи компілятора.
8. Навчання структурі проєктів.
9. Робота з тест-сьютами.
10. Формування правильного стилю кодування.
11. Навички роботи з C++.
12. Застосування стандартних бібліотек.
13. Використання файлів для логування.
14. Навчання unit-тестуванню.
15. Формування грамотної структури коду.
16. Відпрацювання роботи з об’єктами.
17. Реалізація звукових сигналів.
18. Автоматичне тестування програм.
19. Робота з файловою системою.
20. Робота з проєктами у правильних директоріях.
21. Критичне мислення.
22. Аналітичне мислення.
23. Логіка та алгоритміка.
24. Структурованість у роботі.
25. Вирішення нестандартних задач.
26. Повага до викладача.
27. Виконання навчального плану.
28. Підвищення рейтингу.
29. Уникнення перескладання.
30. Успішне складання сесії.
31. Підготовка до роботи в ІТ.
32. Практика для майбутніх співбесід.
33. Портфоліо програміста.
34. Демонстрація відповідальності.
35. Розвиток soft-skills.
36. Розуміння структури тестів.
37. Автоматизація перевірки коду.
38. Підготовка до професійного QA.
39. Застосування логічних перевірок.
40. Тестування в умовах реального проєкту.
41. Організація структури папок.
42. Пошук і виправлення помилок шляху.
43. Навчання перенесенню проєктів.
44. Досвід із роботою в IDE.
45. Робота з різними типами файлів.
46. Підвищення середнього балу.
47. Позитивна характеристика.
48. Успішний захист роботи.
49. Вчасне закриття предмету.
50. Мотивація для подальших досягнень.
51. Відчуття завершення.
52. Досягнення поставленої цілі.
53. Самореалізація.
54. Гордість за себе.
55. Боротьба з прокрастинацією.
56. Створення програмного модуля.
57. Побудова архітектури класу.
58. Робота з includes та namespaces.
59. Практика рефакторингу.
60. Використання шаблонів коду.
61. Вивчення основ ООП у реальній задачі.
62. Розуміння інкапсуляції та поліморфізму.
63. Використання заголовкових файлів.
64. Написання методу в класі.
65. Підключення зовнішніх модулів.
66. Виявлення синтаксичних помилок.
67. Вивчення логів компіляції.
68. Усвідомлення важливості шляхів у проєктах.
69. Навчання налагодженню (debugging).
70. Прийом самостійного пошуку помилок.
71. Щоб не чути 100 звукових сигналів.
72. Щоб не писати "порушено!" у звіті.
73. Щоб не лякати одногрупників компіляцією.
74. Щоб викладач не підозрював змову.
75. Щоб не пояснювати, чому “не було часу”.
76. Підготовка до стажування.
77. Практика створення власного ПЗ.
78. Навички командної розробки.
79. Уміння писати технічну документацію.
80. Участь у хакатонах.
81. Навчання написанню коментарів.
82. Формування правильного README.
83. Створення зрозумілого API класу.
84. Підготовка до open-source розробки.
85. Стандартизація коду.
86. Робота з Git або іншими VCS.
87. Дотримання структурованих папок.
88. Зв’язок тестів і логіки програми.
89. Імітація умов продакшну.
90. Підготовка до участі в командних проєктах.
91. Вчить розподіляти час.
92. Допомагає уникнути авралів.
93. Дає змогу зосередитися на важливому.
94. Допомагає привчитися до дедлайнів.
95. Формує відповідальність.
96. Дає задоволення від вирішення задачі.
97. Надихає на інші проєкти.
98. Розвиває дисципліну.
99. Підвищує впевненість.
100. Просто треба зробити!

Висновок: У результаті виконання лабораторної роботи було створено заголовковий файл ModulesKondratenko.h, який містить клас ClassLab12\_Kondratenko, що формально представляє об’єкт предметної області відповідно до варіанта задачі 12.1. Також розроблено додаток Teacher, який перевіряє коректність розташування файлу main.cpp, генерує 100 звукових сигналів при помилковій структурі проєкту або, навпаки, здійснює unit-тестування об’єкта за тест-сюїтами, зберігаючи результати у файл \Lab12\TestSuite\TestResults.txt.

Ця лабораторна робота дозволила закріпити навички об'єктно-орієнтованого програмування, роботи з файлами, створення і використання модулів, а також основи автоматичного тестування. Отримані знання є важливою складовою професійної підготовки фахівця з ІТ.