

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
* **TestCaseReader** - клас для читання тест-кейсів з файлів
* **TestRunner** - клас для виконання тестів та протоколювання результатів

### 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