Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ №A[12]

з навчальної дисципліни

“Базові методології та технології програмування”

ПРОГРАМНА РЕАЛІЗАЦІЯ АБСТРАКТНИХ ТИПІВ ДАНИХ

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 КБ-24

Заріцкий В. А.

ПЕРЕВІРИВ

викладач кафедри кібербезпеки   
та програмного забезпечення

Коваленко А. С.

Кропивницький – 2025

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 №**1

**Завдання до лабораторної роботи**

1. Як складову заголовкового файлу ModulesZaritsky.h розробити клас ClassLab12\_Zaritsky — 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\_Zaritsky із заголовкового файлу ModulesZaritsky.h та виконує його unit-тестування за тест-сьютом(ами) із \Lab12\TestSuite\, протоколюючи результати тестування в текстовий файл \Lab12\TestSuite\TestResults.txt.

# Аналіз вимог до програмного модуля (класу C++)

Необхідно створити клас ClassLab12\_Zaritsky, який буде абстракцією кубічного кавуна з такими вимогами:

1. Атрибути класу:

* Довжина грані куба (a)
* Колір шкірки кавуна (зовнішній вигляд)
* Колір м'якоті кавуна (внутрішній вигляд)
* Стиглість кавуна (у відсотках)

1. Операції:

* Конструктор(и) для ініціалізації атрибутів
* Надання значень атрибутів (getters)
* Обчислення і повернення об'єму кавуна за формулою
* Зміна значень атрибутів (setters) з валідацією

1. Обмеження і валідація:

* Довжина грані повинна бути додатним числом
* Стиглість має бути в діапазоні від 0% до 100%

# Проєктування архітектури програмного модуля

Клас ClassLab12\_Zaritsky матиме наступну структуру:

Приватні дані-члени:

* edgeLength: double — Довжина грані куба (м)
* string skinColor — Колір шкірки кавуна
* string fleshColor — Колір м'якоті кавуна
* double ripeness — Стиглість кавуна (%)

Публічні функції-члени:

* ClassLab12\_Zaritsky() — конструктор за замовчуванням
* ClassLab12\_Zaritsky(double edge, std::string skin, std::string flesh, double ripe) — конструктор з параметрами
* double getEdgeLength() — повертає довжину грані куба
* string getSkinColor() — повертає колір шкірки
* string getFleshColor() — повертає колір м'якоті
* double getRipeness() — повертає рівень стиглості (у відсотках)
* double getVolume() — повертає/обчислює об'єм куба
* void setEdgeLength(double edge) — встановлює довжину грані куба
* void setFleshColor(std::string flesh) — встановлює колір м’якоті
* void setRipeness(double ripe) — встановлює рівень стиглості

# Реалізація класу ClassLab12\_Zaritsky в заголовковому файлі

#ifndef MODULES\_ZARITSKY\_H

#define MODULES\_ZARITSKY\_H

#include <string>

#include <cmath>

#include <stdexcept>

// Клас, що представляє абстракцію кавуна кубічної форми

class ClassLab12\_Zaritsky {

private:

double edgeLength; // Довжина грані куба (м)

std::string skinColor; // Колір шкірки кавуна

std::string fleshColor; // Колір м'якоті кавуна

double ripeness; // Стиглість кавуна (%)

public:

// Конструктор за замовчуванням

ClassLab12\_Zaritsky();

ClassLab12\_Zaritsky(double edge, std::string skin, std::string flesh, double ripe);

// Отримання довжини грані куба

double getEdgeLength() const;

// Отримання кольору шкірки

std::string getSkinColor() const;

// Отримання кольору м'якоті

std::string getFleshColor() const;

// Отримання рівня стиглості

double getRipeness() const;

// Обчислення об'єму куба

double getVolume() const;

// Встановлення довжини грані куба

void setEdgeLength(double edge);

// Встановлення кольору шкірки

void setSkinColor(std::string skin);

// Встановлення кольору м'якоті

void setFleshColor(std::string flesh);

// Встановлення рівня стиглості

void setRipeness(double ripe);

};

inline ClassLab12\_Zaritsky::ClassLab12\_Zaritsky()

: edgeLength(0.2), skinColor("green"), fleshColor("red"), ripeness(90.0) {

}

inline ClassLab12\_Zaritsky::ClassLab12\_Zaritsky(double edge, std::string skin, std::string flesh, double ripe) {

setEdgeLength(edge);

setSkinColor(skin);

setFleshColor(flesh);

setRipeness(ripe);

}

inline double ClassLab12\_Zaritsky::getEdgeLength() const {

return edgeLength;

}

inline std::string ClassLab12\_Zaritsky::getSkinColor() const {

return skinColor;

}

inline std::string ClassLab12\_Zaritsky::getFleshColor() const {

return fleshColor;

}

inline double ClassLab12\_Zaritsky::getRipeness() const {

return ripeness;

}

inline double ClassLab12\_Zaritsky::getVolume() const {

return pow(edgeLength, 3);

}

inline void ClassLab12\_Zaritsky::setEdgeLength(double edge) {

if (edge <= 0) {

throw std::invalid\_argument("Довжина грані має бути додатним числом");

}

edgeLength = edge;

}

inline void ClassLab12\_Zaritsky::setSkinColor(std::string skin) {

skinColor = skin;

}

inline void ClassLab12\_Zaritsky::setFleshColor(std::string flesh) {

fleshColor = flesh;

}

inline void ClassLab12\_Zaritsky::setRipeness(double ripe) {

if (ripe < 0 || ripe > 100) {

throw std::invalid\_argument("Стиглість має бути в діапазоні від 0 до 100");

}

ripeness = ripe;

}

#endif // MODULES\_ZARITSKY\_H

# Аналіз і постановка задачі завдання 2 (додаток Teacher)

Мета: Розробити консольний додаток Teacher, який виконуватиме автоматизоване тестування класу ClassLab12\_Zaritsky та протоколюватиме результати.

Функціональні вимоги:

* Перевірка розташування файлу проєкту: Додаток має перевіряти, чи файл main.cpp під час компіляції знаходився у теці \Lab12\prj. Якщо умова не виконана, додаток повинен видати 100 звукових сигналів та записати рядок "Встановлені вимоги порядку виконання лабораторної роботи порушено!" у текстовий файл TestResults.txt.
* Unit-тестування класу ClassLab12\_Zaritsky: Якщо файл main.cpp знаходиться у \Lab12\prj, додаток створює об'єкт класу ClassLab12\_Zaritsky із заголовкового файлу ModulesZaritsky.h. Виконує unit-тестування цього об'єкта за тест-сьютами із теки \Lab12\TestSuite\. Протоколює результати тестування у текстовий файл \Lab12\TestSuiteTestResults.txt.

Формат тест-кейсів: Для додатку Teacher я розробив спеціальний формат тест-кейсів, де кожен кейс має наступну структуру:

Test Case ID: TC\_XXX

Action: Опис дії, що тестується

Expected Result: Очікуваний результат

Test Result: PASS/FAIL

Цей формат дозволяє легко парсити текстові файли з тест-кейсами та обробляти їх у коді. Кожен елемент розділяється новим рядком, а різні тест-кейси можуть бути розділені порожнім рядком для кращої читабельності.

Протокол читання тест-кейсів реалізовано в класі TestSuiteProcessor, який аналізує файл рядок за рядком, виявляє ключові слова і формує структури даних для подальшої обробки і виконання тестів.

# Аналіз вимог до програмного засобу Teacher, проєктування архітектури й детальне проєктування ПЗ

Аналіз вимог: Додаток Teacher повинен виконувати дві основні функції: перевірку структури файлів та автоматизоване тестування. Кожна з цих функцій має свої особливості та вимоги до реалізації.

Перевірка структури: Потребує доступу до файлової системи для визначення шляху до файлу main.cpp.

Тестування: Потребує читання тест-кейсів з файлів, створення об'єкта класу, виклику його методів та порівняння отриманих результатів з очікуваними. Це передбачає парсинг текстових файлів та динамічну роботу з об'єктами тестування.

Проєктування архітектури: Додаток Teacher буде мати модульну архітектуру, яка дозволить розділити логіку на окремі компоненти для кращої керованості та розширюваності.

Основні компоненти:

* Функція main: Точка входу в додаток. Відповідає за початкові перевірки (шлях компіляції) та координацію роботи інших компонентів.
* PathChecker (клас/функція): Відповідає за перевірку коректності розташування файлу main.cpp.
* Logger (клас): Відповідає за запис результатів у текстові файли (TestResults.txt та TestSuiteTestResults.txt) та виведення повідомлень користувачеві.
* TestCaseParser (локальний клас або функція): Відповідає за читання та парсинг тестових кейсів з текстових файлів. Буде реалізований як локальний клас або функція, як зазначено в методичних вказівках.
* UnitTestRunner (клас): Координує виконання тестів. Створює об'єкт класу ClassLab12\_Zaritsky, викликає методи тестування та порівнює результати.
* SoundPlayer (клас/функція): Відповідає за відтворення звукових сигналів.