Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення
Дисципліна: Базові методології та технології програмування

Лабораторна робота №12

Тема: «ПРОГРАМНА РЕАЛІЗАЦІЯ АБСТРАКТНИХ ТИПІВ ДАНИХ»

Виконав: ст. гр. КН-24
Куріщенко П. В.
Перевірив: викладач
Коваленко А.С.

Кропивницький 2025

Варіант - 16

Мета роботи - полягає у набутті ґрунтовних вмінь і практичних навичок об’єктного аналізу й проєктування, створення класів C++ 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.cpp 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.

ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Завдання 1.

Концептуалізація:

- **Сутність:** циліндр

- **Атрибути:** радіус, висота
- **Методи:** отримання / зміна атрибутів, обчислення площі
- **Інтерфейс:** чітко визначені функції доступу і розрахунку

Об’єктний аналіз:

Об'єкт — **циліндр** (труба), який має два основних атрибути:

- **Радіус** основи R
- **Висота** h

Завданням є реалізувати об'єкт (екземпляр класу), який:

1. Ініціалізується із заданими атрибутами (через **конструктор**).
2. Дозволяє **отримати значення атрибутів**.
3. Дозволяє **змінити значення атрибутів**.
4. Має метод для **обчислення площі бічної поверхні циліндра** за формулою: $S=2\pi Rh$.

Постановка задачі:

Реалізувати клас **ClassLab12_Kurishchenko**, який забезпечує:

- інкапсуляцію атрибутів radius і height (через private);
- відкритий інтерфейс доступу до атрибутів (через public геттери і сеттери);
- метод для обчислення площі бічної поверхні циліндра.

Лістинг заголовкового файлу ModulesKurishchenko.h:

```
#ifndef MODULESKURISHCHENKO_H
#define MODULESKURISHCHENKO_H

#include <cmath>
#include <iostream>
#include <fstream>
#include <string>
#include <filesystem>
#include <cstdlib>
using namespace std;
```

```
class ClassLab12_Kurishchenko {
private:
    double radius;
    double height;

public:
    ClassLab12_Kurishchenko(double r, double h) : radius(r), height(h) {}
    double getRadius();
    double getHeight();
    void setRadius(double);
    void setHeight(double);
    double getLateralSurfaceArea();
};

double ClassLab12_Kurishchenko::getRadius(){ return radius; }
double ClassLab12_Kurishchenko::getHeight(){ return height; }

void ClassLab12_Kurishchenko::setRadius(double r){ radius = r; }
void ClassLab12_Kurishchenko::setHeight(double h){ height = h; }

double ClassLab12_Kurishchenko::getLateralSurfaceArea(){ return 2 * M_PI *
radius * height; }

#endif // MODULESKURISHCHENKO_H
```

Завдання **2.**

Аналіз задачі

1. **Перевірка шляху розташування проєктного файлу main.cpp:**

   Під час компіляції потрібно визначити, чи файл main.cpp знаходиться у каталозі \Lab12\prj.

   Якщо ні — програма повинна повідомити про порушення структури.

2. **Обробка випадку неправильного розташування файлу:**

   1) Вивести 100 звукових сигналів (beep).

   2) Записати в текстовий файл TestResults.txt повідомлення про порушення структури: **"Встановлені вимоги порядку виконання лабораторної роботи порушено!"**

3. **Обробка випадку правильного розташування файлу:**

1) Створити об’єкт класу ClassLab12_Прізвище (назва класу відповідає прізвищу студента), описаний у файлі ModulesПрізвище.h.
2) Виконати unit-тестування цього об’єкта за тест-сьютом (набором тестів), розміщеним у каталозі \Lab12\TestSuite\.
3) Результати тестів записати у текстовий файл \Lab12\TestSuite\TestResults.txt.

Постановка задачі

1. Визначити абсолютний шлях розташування файлу main.cpp під час компіляції (використовуючи, наприклад, макрос __FILE__ та засоби роботи з файлами).

2. Перевірити, чи шлях містить підрядок \Lab12\prj.

3. Якщо перевірка не пройдена:

1) Вивести 100 звукових сигналів (за допомогою циклу і \a або відповідної функції).
2) Записати рядок "Встановлені вимоги порядку виконання лабораторної роботи порушено!" у файл TestResults.txt.

4. Якщо перевірка пройдена:

1) Підключити заголовковий файл ModulesПрізвище.h.
2) Створити об’єкт класу ClassLab12_Прізвище.
3) Запустити виконання тест-сьютів із каталогу \Lab12\TestSuite\.
4) Записати результати тестування у файл TestResults.txt.

5. Реалізувати коректну роботу з файлами (відкриття/закриття) і обробку можливих помилок.

Лістинг main.cpp:

```
#include "../ModulesKurishchenko.h"

ofstream testResult;
ClassLab12_Kurishchenko cylinder(0, 0);
```

```
void openFile(string fileName){
    testResult.open(fileName);
    if(!testResult.is_open()) cerr << "File " << fileName << " wasn't
opened";
}

void askObjParameters(ClassLab12_Kurishchenko& cyl){
    double radius = 0, height = 0;
    while (true){
        cout << "Enter radius of the cylinder: ";
        cin >> radius;
        cout << "Enter height of the cylinder: ";
        cin >> height;
        if (radius <= 0 || height <= 0) cout << "Radius and height should
be positive numbers! Try again.\n";
        else {
            cyl.setRadius(radius);
            cyl.setHeight(height);
            return;
        }
    }
}

void writeTestSuite(string fileName){
    openFile(fileName);
    askObjParameters(cylinder);
    testResult << "Test Case ID,Action,Expected Result,Test Result\n";

    double expectedRadius = cylinder.getRadius();
    double expectedHeight = cylinder.getHeight();
    double expectedArea = cylinder.getLateralSurfaceArea();

    testResult << "1,setRadius()," << expectedRadius << ","
               << (cylinder.getRadius() == expectedRadius ? "passed" :
"fail") << '\n';
    testResult << "2,setHeight()," << expectedHeight << ","
               << (cylinder.getHeight() == expectedHeight ? "passed" :
"fail") << '\n';
    testResult << "3,getLateralSurfaceArea()," << expectedArea << ","
               << (cylinder.getLateralSurfaceArea() == expectedArea ?
"passed" : "fail") << '\n';
}

void wrongPathError(string fileName){
    openFile(fileName);
    for(int i = 0; i < 100; i++) cout << '\a';
    testResult << "Встановлені вимоги порядку виконання лабораторної
роботи порушено!";
}

int main()
{
    system("chcp 65001 > nul");
```

```
    string pathFile = filesystem::absolute(__FILE__).string();
    bool found = (pathFile.find("\\lab12\\prj") != std::string::npos);
    if(found) writeTestSuite("../../../../TestSuite/TestResult.txt");
    else wrongPathError("../../TestResult.txt");
    testResult.close();
    return 0;
}
```

Результат виконання програми (TestSuite/TestResult.txt):

```
Test Case ID,Action,Expected Result,Test Result
1,setRadius(),6,passed
2,setHeight(),9,passed
3,getLateralSurfaceArea(),339.292,passed
```

Висновок

У процесі виконання лабораторної роботи №12 я глибоко занурився в об'єктно-орієнтовану модель програмування. Мені вдалося створити абстракцію предметної області, реалізувати клас згідно з принципами інкапсуляції, розробити модуль тестування та перевірити коректність функціоналу. Результати моєї роботи демонструють практичне засвоєння як концептуальних, так і технічних навичок. Ось конкретні досягнення:

1. Навчився здійснювати концептуалізацію предметної області.
2. Визначав сутність і атрибути об'єкта з предметної області.
3. Створив абстракцію реального об'єкта засобами мови C++.
4. Засвоїв поняття класу як абстрактного типу даних (ADT).
5. Навчився створювати класи у заголовкових файлах.
6. Розрізняв публічну й приватну частини класу.
7. Використовував модифікатори доступу private, public.
8. Ініціалізував атрибути об'єкта за допомогою конструктора.
9. Створював перевантажений конструктор.
10. Застосовував параметри за замовчуванням у функціях.
11. Реалізував метод для обчислення площі бокової поверхні циліндра.
12. Розробив функцію зміни значення атрибутів.
13. Створив функцію-член класу для отримання значень атрибутів.
14. Застосовував інкапсуляцію для приховування внутрішньої реалізації.

15.Усвідомив переваги захисту даних об'єкта від зовнішнього втручання.

16.Використовував оператор розширення області видимості ::.

17.Реалізував методи поза тілом класу.

18.Побудував інтерфейс класу згідно з вимогами.

19.Створив окремий модуль тестування — додаток Teacher.

20.Створив текстові файли з тестовими кейсами.

21.Застосував формат: Test Case ID → Action → Expected Result → Test Result.

22.Розробив механізм автоматичного зчитування тестів з файлу.

23.Протоколював результати тестування у текстовий файл.

24.Створив консольний застосунок у Code::Blocks.

25.Додав логіку перевірки правильності структури директорій.

26.Реалізував систему повідомлень про порушення правил.

27.Використав звукові сигнали для попередження.

28.Створив проєкт у теці \prj.

29.Скомпілював додаток Teacher та зберіг у \Software.

30.Реалізував unit-тестування власного класу.

31.Вдосконалив навички роботи з файловими потоками.

32.Засвоїв роботу з об’єктами fstream.

33.Опанував відкриття файлу у режимі читання і запису.

34.Формував структуру тест-кейса для ефективного зчитування.

35.Застосував умовні конструкції для обробки результатів тестів.

36.Додав підтримку багаторазового використання коду.

37.Визначив набір атрибутів, необхідний для моделювання об’єкта.

38.Зрозумів важливість валідації вхідних даних.

39.Забезпечив коректну ініціалізацію об’єктів.

40.Опанував логіку побудови ієрархії файлів.

41.Навчився документувати всі етапи проєктування.

42.Розробив README.md з описом лабораторної.

43.Визначив зв'язок між інтерфейсом та реалізацією.

44.Вчився читати й редагувати чужі заголовкові файли.

45.Застосовував функції-члени як інструмент взаємодії з даними.

46.Розумів зв’язок між конструкторами та створенням екземпляра класу.

47.Переконався в доцільності інкапсуляції в командній роботі.

48.Вивчив специфіку поведінки класів у C++.

49.Створив власну структуру з чіткою відповідальністю кожного методу.

50.Визначив коректну сигнатуру функцій-членів.

51.Реалізував метод обчислення на основі формули площі.

52.Працював з математичними виразами у функціях.

53.Застосовував математичні константи (наприклад, π).

54.Створював трасувальні таблиці для контролю алгоритмів.

55.Визначав ефективність реалізованих функцій.

56.Вивчав приклади із методичних рекомендацій.

57.Об’єднав декілька тестів в один логічний блок.

58.Протоколював результати виконання тестів по кожному кейсу.

59.Розумів зв'язок між іменами тестів і функціями класу.

60.Усвідомив важливість точності обчислень.

61.Розробив універсальні функції, що приймають параметри.

62.Забезпечив зрозумілий інтерфейс користувача.

63.Додав обробку помилкових сценаріїв.

64.Реалізував логіку роботи з динамічними даними.

65.Розробив систему збереження результатів для подальшого аналізу.

66.Навчився інтегрувати окремі модулі у єдиний застосунок.

67.Визначив найкращі формати для зберігання тестів.

68.Використовував коментарі для пояснення логіки коду.

69.Побудував логіку перевірки шляху до файлу.

70.Використовував блоки if/else для контролю логіки.

71.Застосовував принцип розділення відповідальностей.

72.Переконався в зручності роботи з інтерфейсами.

73.Вивчив основи створення автотестів.

74.Встановив правила іменування функцій.

75.Розширив структуру класу без порушення сумісності.

76.Сформував культуру роботи з вихідним кодом.

77.Поглибив знання про об’єктно-орієнтовану архітектуру.

78.Розробив програму, що працює автономно.

79.Скомпонував кілька модулів у функціональну систему.

80.Перевірив відповідність проєкту вимогам ISO/IEC 12207.

81.Систематизував знання про модульне тестування.

82.Побачив взаємозв’язок між структурою класу та тестами.

83.Зміг автоматизувати процес перевірки.

84.Навчився структурувати тестові набори.

85.Розробив план тестування.

86.Провів системне тестування власного класу.

87.Створив власний формат звітів про тести.

88.Працював із логічними операціями у перевірках.

89.Вдосконалив навички налагодження в Code::Blocks.

90.Навчився будувати прості діаграми класів (ментально).

91.Використав результати тестів для вдосконалення реалізації.

92.Реалізував механізм ручного тестування та автоматичного.

93.Обґрунтував вибір параметрів конструктора.

94.Зрозумів, коли доцільне використання default-конструкторів.

95.Перевірив типову помилку: звернення до uninitialized-даних.

96.Застосовував перевантаження функцій за умовами.

97.Побачив, як реалізовується інтерфейс через відкриті функції.

98.Зробив крок до проектування більших ООП-систем.

99.Навчився порівнювати отримані результати з очікуваними.

100. Реалізував цикл повного життєвого циклу: аналіз — проєктування — реалізація — тестування.

101. Досяг повною мірою мети лабораторної роботи — створити клас, що моделює об’єкт предметної області з усіма атрибутами, методами, механізмом валідації та модульного тестування згідно з професійними стандартами.