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

ЗАВДАННЯ ВИДАВ
доцент кафедри кібербезпеки
та програмного забезпечення
Доренський О. П.
https://github.com/odorenskyi/

ВИКОНАВ
студент академічної групи КБ-24
Олефіров Г.Є.
https://github.com/GlibOlefirov

ПЕРЕВІРИВ
ст. викладач кафедри кібербезпеки
та програмного забезпечення
Коваленко А.С.

Кропивницький – 2025

# ПРОГРАМНА РЕАЛІЗАЦІЯ АБСТРАКТНИХ ТИПІВ ДАНИХ

**Мета:** 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.

**Варіант 38**

## 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.

## Концептуалізація предметної області

**Предметна область**: поверхня стола у формі еліпса. Відповідні ключові поняття (концепти):

- **Еліпс** – геометрична фігура, що описує форму стільниці.
- **Велика піввісь (a)** – максимальний радіус еліпса.
- **Мала піввісь (b)** – мінімальний радіус еліпса.
- **Площа (S)** – площа поверхні еліпса, яку треба обчислити.

## Аналіз та постановка задачі

**Мета**: написати C++-клас, який інкапсулює параметри еліптичної поверхні столу й забезпечує коректне збереження їх, а також обчислення площі поверхні.

### Вхідні дані

a — довжина великої піввісі (double, >0)

b — довжина малої піввісі (double, >0)

### Вихідні дані

S — площа еліпса:

$S=\pi \cdot a \cdot b$

## Об’єктний аналіз

Виділимо єдину сутність предметної області:

| Сутність | Атрибути | Операції (поведінка) |
|---|---|---|
| **EllipseTable** | • a – довжина великої піввісі (double)<br>• b – довжина малої піввісі (double) | • `getA()`, `getB()` – читання параметрів<br>• `setA(a)`, `setB(b)` – встановлення (з валідацією >0)<br>• `area()` – обчислення площі: S = π·a·b |

**Задача 12.1.** Розробити клас `ClassLab12_<Прізвище>`, що моделює стільницю у формі еліпса. Клас повинен:

1. Ініціалізувати параметри `a` та `b` через конструктор.
2. Забезпечити публічні методи доступу (геттери й сеттери) із валідацією.
3. Реалізувати метод `area()`, який повертає площу еліптичної поверхні.
4. Забезпечити інкапсуляцію внутрішнього представлення.

## Лістинг ModulesOlefirov.h

```
#ifndef MODULESOLEFIROV_H_INCLUDED

#define MODULESOLEFIROV_H_INCLUDED

#include <string>

#include <stdexcept>

 #include <cmath>

double calculate_S(double x, double y, double z);

void findShoeSize();

void processNumber();

void calculateGasPayment();

#include // для std::string

// Функція для задачі 10.1 – обробка слова з тексту void processText(const std::string&
inputFile, const std::string& outputFile);

// Функція для задачі 10.2 – дописати кількість символів і дату void appendInfoToFile(const
std::string& filename);

// Функція для задачі 10.3 – обчислення значення та вивід двійкового b void
processTask10_3(double x, double y, double z, int b, const std::string& outputFilename);

class ClassLab12_Olefirov {
```

```
private: double a; // велика піввісь

double b; // мала піввісь

public: // Конструктор за замовчуванням

ClassLab12_Olefirov() { a = 1.0; b = 1.0; }

// Конструктор з параметрами
ClassLab12_Olefirov(double a_, double b_) {
    setA(a_);
    setB(b_);
}

// Гетер для a
double getA() const {
    return a;
}

// Гетер для b
double getB() const {
    return b;
}

// Сетер для a з перевіркою
void setA(double a_ = 1.0) {
    if (a_ <= 0)
        throw std::invalid_argument("Параметр a має бути додатнім.");
    a = a_;
}

// Сетер для b з перевіркою
void setB(double b_ = 1.0) {
    if (b_ <= 0)
        throw std::invalid_argument("Параметр b має бути додатнім.");
    b = b_;
}

// Обчислення площі еліпса
double area() const {
    return M_PI * a * b;
}


}; #endif // MODULESOLEFIROV_H_INCLUDED
```

## Аналіз і постановка задачі Teacher

**Teacher** — консольний застосунок для виконання Unit-тестування об’єкта класу ClassLab12_Прізвище згідно з підготовленими тест-кейсами.

## Мета програми

Автоматизувати:

- перевірку коректності структури проєкту (розміщення в каталозі \Lab12\prj);
- виконання unit-тестування розробленого класу;
- протоколювання результатів тестування у файл.

Програма Teacher виконує роль контролера якості — забезпечує відповідність структури проєкту, правильність логіки реалізації класу та документує результати автоматичного тестування.

## Результати тестування

## (вміст TestResult)

Тестування файлу:test1.txt

Test Case ID: T1 | Action: setA | Expected: 5 | Result: passed

Test Case ID: T2 | Action: setB | Expected: 3 | Result: passed

Test Case ID: T3 | Action: area | Expected: 47.1 | Actual: 47.1239 | Result: passed

Test Case ID: T4 | Action: setA | Expected: 0 | ВИНЯТОК: Параметр a має бути додатнім.

Test Case ID: T5 | Action: setB | Expected: -1 | ВИНЯТОК: Параметр b має бути додатнім.

Test Case ID: T6 | Action: setA | Expected: 2 | Result: passed

Test Case ID: T7 | Action: setB | Expected: 4 | Result: passed

Test Case ID: T8 | Action: area | Expected: 25.1 | Actual: 25.1327 | Result: passed

Тестування файлу:test2.txt

Test Case ID: T1 | Action: setA | Expected: 10 | Result: passed

Test Case ID: T2 | Action: setB | Expected: 5 | Result: passed

Test Case ID: T3 | Action: area | Expected: 157 | Actual: 157.08 | Result: passed

Test Case ID:  | Action:  | Expected: 157 | Unknown action!

Test Case ID: T4 | Action: setA | Expected: 3.5 | Result: passed

Test Case ID: T5 | Action: setB | Expected: 2.5 | Result: passed

Test Case ID: T6 | Action: area | Expected: 27.5 | Actual: 27.4889 | Result: passed

Test Case ID:  | Action:  | Expected: 27.5 | Unknown action!

Test Case ID: T7 | Action: setA | Expected: -2 | ВИНЯТОК: Параметр a має бути додатнім.

Test Case ID: T8 | Action: setB | Expected: 0 | ВИНЯТОК: Параметр b має бути додатнім.

**Висновок**

1. Здобуття практичних навичок моделювання об’єктів предметної області.
2. Формалізація абстракцій через C++-класи.
3. Вивчення принципу інкапсуляції даних.
4. Розуміння поняття інформаційного приховування.
5. Ознайомлення з конструктором за замовчуванням.
6. Засвоєння перевантаження конструкторів.
7. Практика роботи з методами-селекторами (геттерами).
8. Практика роботи з методами-мутаторами (сеттерами).
9. Реалізація перевірки вхідних параметрів у сеттерах.
10. Навчання викидати та обробляти виключення.
11. Вивчення використання стандартного винятка std::invalid_argument.
12. Обчислення математичних виразів у методах-членах.
13. Розрахунок площі еліпса з використанням M_PI.
14. Практика підключення заголовочних файлів.
15. Ознайомлення з препроцесорними директивами.
16. Використання макросів для забезпечення кросплатформеності.
17. Вивчення структури проєктів у Code::Blocks.
18. Формування структури каталогів Lab12\prj, TestSuite тощо.
19. Практика роботи з файловим введенням/виведенням.
20. Використання класу std::ifstream для читання файлів.
21. Використання класу std::ofstream для запису результатів.
22. Навички обробки рядкових даних через std::stringstream.
23. Формування та парсинг тест-кейсу з рядка.
24. Розробка власного тест-сьюту для класу.

25. Автоматизація тестування через цикл викликів.
26. Формування єдиного файлу результатів TestResults.txt.
27. Практика порівняння очікуваних і фактичних значень.
28. Використання умови для PASS/FAIL з точністю.
29. Розробка чіткого та зрозумілого формату протоколу.
30. Практика обробки невідомих дій (Unknown action).
31. Вміння додавати діагностичні повідомлення в протокол.
32. Навички налагодження шляху до файлів із GetModuleFileNameA.
33. Розуміння ролі робочої директорії програми.
34. Прийоми визначення абсолютного та відносного шляху.
35. Практика створення директорій через std::filesystem (за бажанням).
36. Вивчення можливостей бібліотеки <filesystem>.
37. Засвоєння синтаксису простору імен std.
38. Практика обробки помилок відкриття файлів.
39. Використання std::cin.get() для утримання консолі.
40. Формування зручного інтерфейсу користувача в консолі.
41. Поглиблення знань про цикл while (std::getline).
42. Відпрацювання роботи з масивом рядків для кількох тестів.
43. Навички роботи з динамічним створенням імен файлів.
44. Розуміння важливості кодування UTF-8 без BOM.
45. Вивчення принципів кросплатформеності: Windows vs Linux.
46. Підготовка до використання Mac OS X у розробці.
47. Ознайомлення з Git-репозиторіями та структурою project.
48. Практика документування коду в README.md.
49. Виготовлення протоколу виконання лабораторної роботи.
50. Формування звіту у форматі згідно з ДСТУ 3008:2015.
51. Навички оформлення технічної документації.
52. Практика аналізу вимог програмного модуля.
53. Розробка архітектури програмного класу.
54. Детальне проєктування інтерфейсів методів.
55. Формування специфікації кожного методу класу.
56. Одержання артефактів аналізу й проєктування.

57. Оформлення артефактів у звіті лабораторної роботи.
58. Виконання модульного тестування об’єктів класу.
59. Навички написання тест-кейсу для кожної функції.
60. Засвоєння короткої структури тест-кейсу: ID → Action → Expected → Result.
61. Формування єдиного місця зберігання тест-сьютів.
62. Розуміння принципів побудови ADT (Abstract Data Type).
63. Концептуалізація предметної області через UML (за бажанням).
64. Практична реалізація UML-діаграм у коді.
65. Отримання досвіду програмного синтезу класу.
66. Забезпечення початкової ініціалізації через конструктор.
67. Дотримання принципу DRY (Don’t Repeat Yourself).
68. Вправи з розділенням реалізації методів поза тілом класу.
69. Використання оператора розширення області видимості ::.
70. Досвід об’єктного аналізу й проєктування в одній роботі.
71. Практика роботи з макрооператорами під час збірки.
72. Ознайомлення з можливостями препроцесора в C++.
73. Розуміння ролі include-guard в заголовках.
74. Засвоєння специфіки компіляції в IDE Code::Blocks.
75. Налагодження проєкту під Debug та Release конфігурації.
76. Отримання досвіду створення makefile (за бажанням).
77. Контроль версій вихідного коду через Git.
78. Автоматичне збереження результатів тестування.
79. Підвищення якості коду через покриття тестами.
80. Зниження ризику регресії при зміні коду.
81. Формування звички писати коментарі до коду.
82. Вміння структурувати великі проєкти.
83. Практика багатофайлової організації проєкту.
84. Ознайомлення з принципами SOLID (за бажанням).
85. Розуміння ролі інтерфейсів і абстракцій.
86. Забезпечення підтримки та розширюваності коду.
87. Здобуття навичок оцінки результатів тестування.

88. Формування вмінь аналізу причин невдач тестів.
89. Розвиток критичного мислення під час верифікації.
90. Підготовка до сертифікації C++ розробника.
91. Практична підготовка до співбесід на позицію C++.
92. Засвоєння навичок роботи з документацією ISO/IEC 12207.
93. Відпрацювання отримання та оформлення висновків.
94. Формування вміння підготувати конструктор звіту.
95. Розвиток навичок самостійного аналізу виконаної роботи.
96. Отримання аргументів для звіту (самообґрунтування).
97. Засвоєння принципів якісного оформлення лабзвіту.
98. Підготовка до публічного захисту результатів.
99. Отримання досвіду комплексного виконання проєкту.
100. Підвищення загального рівня компетентності з C++.