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”

На тему:

“ Програмна реалізація абстрактних типів даних”

ВИКОНАВ

студент академічної групи

КБ-24

Сумар Ілля

ПЕРЕВІРИВ

викладач Коваленко А. С.

м. Кропивницький 2025

**Мета роботи:**

Мета цієї лабораторної роботи полягає в набутті практичних навичок об'єктно-орієнтованого аналізу та проєктування, а також у розвитку вмінь створення класів C++ та їх тестування. Крім того, робота спрямована на використання препроцесорних директив, макросів і макрооператорів у кросплатформовому середовищі розробки Code::Blocks.

**Кроки виконання роботи**

1. Підготовка до роботи:
   * Завантажити власний Git-репозиторій.
   * Записати мету лабораторної роботи, номер варіанта та завдання до звіту.
2. Налаштування середовища:
   * У папці \Lab12 створити файли README.md, а також теки prj, Software, TestSuite, Report.
3. Об'єктний аналіз та проєктування:
   * Виконати об'єктний аналіз сутності згідно з варіантом завдання.
   * Провести аналіз та постановку задачі 12.1.
   * Програмно реалізувати абстракцію сутності предметної області, враховуючи вимоги стандарту ISO/IEC 12207.
4. Розробка класу C++:
   * Створити клас ClassLab12\_Прізвище згідно з вимогами.
   * Забезпечити початкову ініціалізацію атрибутів конструктором.
   * Реалізувати відкриті функції-члени для доступу до атрибутів класу.
5. Тестування:
   * Розробити тест-сьюти для кожної функції модуля.
   * Виконати модульне тестування об'єкта класу за допомогою додатка Teacher.
6. Реалізація додатка Teacher:
   * Виконати аналіз вимог до програмного засобу Teacher.
   * Запропонувати текстовий формат тест-кейсів.
   * Реалізувати додаток Teacher для читання тест-кейсів та проведення тестування.
7. Документування та звіт:
   * Задокументувати всі отримані артефакти та включити їх до звіту.
   * Підготувати звіт про виконання лабораторної роботи, оформлений згідно з ДСТУ 3008:2015.
8. Захист роботи:
   * Представити до захисту звіт з виконаної лабораторної роботи та проєкт у Git-репозиторії.

**Завдання:**



**Аналіз**

Sumar.exe реалізує клас ClassLab12\_Sumar, який представляє абстракцію лави в лампі. Основні функції класу включають обчислення об'єму лампи, об'єму лави та об'єму рідини. Клас має методи для встановлення та отримання значень радіуса, висоти та відсотка лави.

**Постановка задачі**

1. Створення класу ClassLab12\_Sumar:
   * Реалізувати конструктор класу, який ініціалізує атрибути радіуса, висоти та відсотка лави.
   * Забезпечити методи для встановлення та отримання значень атрибутів з перевіркою на коректність вхідних даних.
2. Обчислення об'ємів:
   * Реалізувати методи для обчислення загального об'єму лампи, об'єму лави та об'єму рідини на основі поточних значень атрибутів.
3. Тестування:
   * Розробити тести для перевірки коректності роботи методів класу.
   * Забезпечити можливість тестування класу за допомогою додатка Teacher, який читає тести з файлу та записує результати в файл TestResults.txt.

**Лістинг розробленої програми Sumar.exe**

#include <iostream>

#include <cmath>

#include <stdexcept>

#include <windows.h>

class ClassLab12\_Sumar {

private:

    double radius;

    double height;

    double lavaPercentage;

public:

    ClassLab12\_Sumar(double r, double h, double lavaPerc)

        : radius(r), height(h), lavaPercentage(lavaPerc) {

        validateInputs(r, h, lavaPerc);

    }

    double getRadius() const { return radius; }

    double getHeight() const { return height; }

    double getLavaPercentage() const { return lavaPercentage; }

    double calculateVolume() const {

        double baseArea = M\_PI \* radius \* radius;

        return baseArea \* height;

    }

    double getLavaVolume() const {

        return calculateVolume() \* lavaPercentage / 100.0;

    }

    double getLiquidVolume() const {

        return calculateVolume() - getLavaVolume();

    }

    void setRadius(double r) {

        if (r > 0) {

            radius = r;

        } else {

            throw std::invalid\_argument("Радіус має бути позитивним числом.");

        }

    }

    void setHeight(double h) {

        if (h > 0) {

            height = h;

        } else {

            throw std::invalid\_argument("Висота має бути позитивним числом.");

        }

    }

    void setLavaPercentage(double lavaPerc) {

        if (lavaPerc >= 0 && lavaPerc <= 100) {

            lavaPercentage = lavaPerc;

        } else {

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

        }

    }

private:

    void validateInputs(double r, double h, double lavaPerc) {

        if (r <= 0 || h <= 0) {

            throw std::invalid\_argument("Радіус та висота мають бути позитивними числами.");

        }

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

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

        }

    }

};

void displayMenu() {

    std::cout << "\nМеню:\n";

    std::cout << "1. Показати об'єм лампи\n";

    std::cout << "2. Показати об'єм лави\n";

    std::cout << "3. Показати об'єм рідини\n";

    std::cout << "4. Змінити радіус\n";

    std::cout << "5. Змінити висоту\n";

    std::cout << "6. Змінити відсоток лави\n";

    std::cout << "7. Вийти\n";

    std::cout << "Оберіть опцію: ";

}

void displayCurrentParameters(const ClassLab12\_Sumar& lamp) {

    std::cout << "Поточні параметри лампи:\n";

    std::cout << "Радіус: " << lamp.getRadius() << "\n";

    std::cout << "Висота: " << lamp.getHeight() << "\n";

    std::cout << "Відсоток лави: " << lamp.getLavaPercentage() << "\n";

}

int main() {

    SetConsoleOutputCP(65001);

    SetConsoleCP(65001);

    std::cout << "Автор: Ілля Сумар\n";

    std::cout << "Формула для обчислення об'єму циліндра: V = π \* r^2 \* h\n\n";

    ClassLab12\_Sumar lamp(5.0, 10.0, 30.0);

    displayCurrentParameters(lamp);

    int choice;

    do {

        displayMenu();

        std::cin >> choice;

        try {

            switch (choice) {

                case 1:

                    std::cout << "Об'єм лампи: " << lamp.calculateVolume() << std::endl;

                    break;

                case 2:

                    std::cout << "Об'єм лави: " << lamp.getLavaVolume() << std::endl;

                    break;

                case 3:

                    std::cout << "Об'єм рідини: " << lamp.getLiquidVolume() << std::endl;

                    break;

                case 4: {

                    double r;

                    std::cout << "Введіть новий радіус: ";

                    std::cin >> r;

                    lamp.setRadius(r);

                    displayCurrentParameters(lamp);

                    break;

                }

                case 5: {

                    double h;

                    std::cout << "Введіть нову висоту: ";

                    std::cin >> h;

                    lamp.setHeight(h);

                    displayCurrentParameters(lamp);

                    break;

                }

                case 6: {

                    double lavaPerc;

                    std::cout << "Введіть новий відсоток лави: ";

                    std::cin >> lavaPerc;

                    lamp.setLavaPercentage(lavaPerc);

                    displayCurrentParameters(lamp);

                    break;

                }

                case 7:

                    std::cout << "Вихід..." << std::endl;

                    break;

                default:

                    std::cout << "Невірний вибір, спробуйте ще раз." << std::endl;

            }

        } catch (const std::exception& e) {

            std::cerr << "Помилка: " << e.what() << std::endl;

            // Програма продовжує роботу після виведення повідомлення про помилку

        }

    } while (choice != 7);

    return 0;

}

**Лістинг розробленої програми Teacher.exe:**

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

#include <stdexcept>

#include <windows.h>

#include <cmath>

using namespace std;

class ClassLab12\_Sumar {

private:

    double radius;

    double height;

    double lavaPercentage;

public:

    ClassLab12\_Sumar(double r, double h, double lavaPerc)

        : radius(r), height(h), lavaPercentage(lavaPerc) {

        validateInputs(r, h, lavaPerc);

    }

    double getRadius() const { return radius; }

    double getHeight() const { return height; }

    double getLavaPercentage() const { return lavaPercentage; }

    double calculateVolume() const {

        double baseArea = M\_PI \* radius \* radius;

        return baseArea \* height;

    }

    double getLavaVolume() const {

        return calculateVolume() \* lavaPercentage / 100.0;

    }

    double getLiquidVolume() const {

        return calculateVolume() - getLavaVolume();

    }

    void setRadius(double r) {

        if (r > 0) {

            radius = r;

        } else {

            throw invalid\_argument("Помилка: Радіус має бути позитивним числом.");

        }

    }

    void setHeight(double h) {

        if (h > 0) {

            height = h;

        } else {

            throw invalid\_argument("Помилка: Висота має бути позитивним числом.");

        }

    }

    void setLavaPercentage(double lavaPerc) {

        if (lavaPerc >= 0 && lavaPerc <= 100) {

            lavaPercentage = lavaPerc;

        } else {

            throw invalid\_argument("Помилка: Відсоток лави має бути в діапазоні від 0 до 100.");

        }

    }

private:

    void validateInputs(double r, double h, double lavaPerc) {

        if (r <= 0 || h <= 0) {

            throw invalid\_argument("Помилка: Радіус та висота мають бути позитивними числами.");

        }

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

            throw invalid\_argument("Помилка: Відсоток лави має бути в діапазоні від 0 до 100.");

        }

    }

};

void runTestSuite() {

    SetConsoleOutputCP(CP\_UTF8);

    system("chcp 65001 > nul");

    ifstream testFile("TestSuite.txt");

    ofstream resultsFile("TestResults.txt");

    if (!testFile.is\_open()) {

        cerr << "Помилка: файл тестів не знайдено." << endl;

        MessageBeep(MB\_ICONERROR);

        return;

    }

    string line;

    while (getline(testFile, line)) {

        istringstream iss(line);

        int testNumber, functionNumber;

        double value;

        string expectedResult;

        iss >> testNumber >> functionNumber >> value;

        getline(iss, expectedResult);

        expectedResult.erase(0, expectedResult.find\_first\_not\_of(" \t\n\r\f\v"));

        ClassLab12\_Sumar lamp(5.0, 10.0, 30.0);

        try {

            switch (functionNumber) {

                case 1: {

                    ostringstream oss;

                    oss << "Об'єм лампи: " << lamp.calculateVolume();

                    if (oss.str() == expectedResult) {

                        resultsFile << "Тест " << testNumber << " з функцією " << functionNumber << ": PASSED, очікувалося: " << expectedResult << ", отримано: " << oss.str() << "\n";

                    } else {

                        resultsFile << "Тест " << testNumber << " з функцією " << functionNumber << ": FAILED, очікувалося: " << expectedResult << ", отримано: " << oss.str() << "\n";

                    }

                    break;

                }

                case 2: {

                    ostringstream oss;

                    oss << "Об'єм лави: " << lamp.getLavaVolume();

                    if (oss.str() == expectedResult) {

                        resultsFile << "Тест " << testNumber << " з функцією " << functionNumber << ": PASSED, очікувалося: " << expectedResult << ", отримано: " << oss.str() << "\n";

                    } else {

                        resultsFile << "Тест " << testNumber << " з функцією " << functionNumber << ": FAILED, очікувалося: " << expectedResult << ", отримано: " << oss.str() << "\n";

                    }

                    break;

                }

                case 3: {

                    ostringstream oss;

                    oss << "Об'єм рідини: " << lamp.getLiquidVolume();

                    if (oss.str() == expectedResult) {

                        resultsFile << "Тест " << testNumber << " з функцією " << functionNumber << ": PASSED, очікувалося: " << expectedResult << ", отримано: " << oss.str() << "\n";

                    } else {

                        resultsFile << "Тест " << testNumber << " з функцією " << functionNumber << ": FAILED, очікувалося: " << expectedResult << ", отримано: " << oss.str() << "\n";

                    }

                    break;

                }

                case 4: {

                    try {

                        lamp.setRadius(value);

                        ostringstream oss;

                        oss << "Поточні параметри лампи: Радіус: " << lamp.getRadius() << " Висота: " << lamp.getHeight() << " Відсоток лави: " << lamp.getLavaPercentage();

                        if (oss.str() == expectedResult) {

                            resultsFile << "Тест " << testNumber << " з функцією " << functionNumber << " та значенням " << value << ": PASSED, очікувалося: " << expectedResult << ", отримано: " << oss.str() << "\n";

                        } else {

                            resultsFile << "Тест " << testNumber << " з функцією " << functionNumber << " та значенням " << value << ": FAILED, очікувалося: " << expectedResult << ", отримано: " << oss.str() << "\n";

                        }

                    } catch (const invalid\_argument& e) {

                        if (string(e.what()) == expectedResult) {

                            resultsFile << "Тест " << testNumber << " з функцією " << functionNumber << " та значенням " << value << ": PASSED, очікувалося: " << expectedResult << ", отримано: " << e.what() << "\n";

                        } else {

                            resultsFile << "Тест " << testNumber << " з функцією " << functionNumber << " та значенням " << value << ": FAILED, очікувалося: " << expectedResult << ", отримано: " << e.what() << "\n";

                        }

                    }

                    break;

                }

                case 5: {

                    try {

                        lamp.setHeight(value);

                        ostringstream oss;

                        oss << "Поточні параметри лампи: Радіус: " << lamp.getRadius() << " Висота: " << lamp.getHeight() << " Відсоток лави: " << lamp.getLavaPercentage();

                        if (oss.str() == expectedResult) {

                            resultsFile << "Тест " << testNumber << " з функцією " << functionNumber << " та значенням " << value << ": PASSED, очікувалося: " << expectedResult << ", отримано: " << oss.str() << "\n";

                        } else {

                            resultsFile << "Тест " << testNumber << " з функцією " << functionNumber << " та значенням " << value << ": FAILED, очікувалося: " << expectedResult << ", отримано: " << oss.str() << "\n";

                        }

                    } catch (const invalid\_argument& e) {

                        if (string(e.what()) == expectedResult) {

                            resultsFile << "Тест " << testNumber << " з функцією " << functionNumber << " та значенням " << value << ": PASSED, очікувалося: " << expectedResult << ", отримано: " << e.what() << "\n";

                        } else {

                            resultsFile << "Тест " << testNumber << " з функцією " << functionNumber << " та значенням " << value << ": FAILED, очікувалося: " << expectedResult << ", отримано: " << e.what() << "\n";

                        }

                    }

                    break;

                }

                case 6: {

                    try {

                        lamp.setLavaPercentage(value);

                        ostringstream oss;

                        oss << "Поточні параметри лампи: Радіус: " << lamp.getRadius() << " Висота: " << lamp.getHeight() << " Відсоток лави: " << lamp.getLavaPercentage();

                        if (oss.str() == expectedResult) {

                            resultsFile << "Тест " << testNumber << " з функцією " << functionNumber << " та значенням " << value << ": PASSED, очікувалося: " << expectedResult << ", отримано: " << oss.str() << "\n";

                        } else {

                            resultsFile << "Тест " << testNumber << " з функцією " << functionNumber << " та значенням " << value << ": FAILED, очікувалося: " << expectedResult << ", отримано: " << oss.str() << "\n";

                        }

                    } catch (const invalid\_argument& e) {

                        if (string(e.what()) == expectedResult) {

                            resultsFile << "Тест " << testNumber << " з функцією " << functionNumber << " та значенням " << value << ": PASSED, очікувалося: " << expectedResult << ", отримано: " << e.what() << "\n";

                        } else {

                            resultsFile << "Тест " << testNumber << " з функцією " << functionNumber << " та значенням " << value << ": FAILED, очікувалося: " << expectedResult << ", отримано: " << e.what() << "\n";

                        }

                    }

                    break;

                }

                default:

                    resultsFile << "Невідомий вибір функції: " << functionNumber << "\n";

            }

        } catch (const exception& e) {

            resultsFile << "Тест " << testNumber << " з функцією " << functionNumber << " та значенням " << value << ": FAILED, помилка: " << e.what() << "\n";

        }

    }

    testFile.close();

    resultsFile.close();

}

int main() {

    runTestSuite();

    cout << "Тести завершено." << endl;

    system("pause");

    return 0;

}

**Аргументи виконання лабораторної роботи:**

1. Розвиток навичок об'єктно-орієнтованого програмування.
2. Зміцнення розуміння принципу інкапсуляції.
3. Набутий досвід у використанні класів C++.
4. Вдосконалення навичок роботи з конструкторами та деструкторами.
5. Розуміння важливості ініціалізації об'єктів.
6. Набуття досвіду у використанні методів класу.
7. Розвиток навичок роботи з атрибутами класу.
8. Вдосконалення навичок роботи з вказівниками та посиланнями.
9. Розуміння принципу спадкування.
10. Набуття досвіду у використанні віртуальних функцій.
11. Розвиток навичок роботи з шаблонами класів.
12. Вдосконалення навичок роботи з операторами.
13. Розуміння принципу поліморфізму.
14. Набутий досвід у використанні абстрактних класів.
15. Розвиток навичок роботи з використанням.
16. Вдосконалення навичок роботи з файлами.
17. Розуміння принципу роботи з динамічною пам'яттю.
18. Набуття досвіду у використанні стандартних контейнерів.
19. Розвиток навичок роботи з алгоритмами STL.
20. Вдосконалення навичок роботи з ітераторами.
21. Розуміння принципу роботи з функціональними об'єктами.
22. Набуття досвіду у використанні лямбда-виразів.
23. Розвиток навичок роботи з макросами.
24. Вдосконалення навичок роботи з препроцесорними директивами.
25. Розуміння принципу роботи з модулями.
26. Набуття досвіду у використанні просторів імен.
27. Розвиток навичок роботи з шаблонами функцій.
28. Вдосконалення навичок роботи з перевантаженням операторів.
29. Розуміння принципу роботи з дружніми функціями.
30. Набуття досвіду у використанні статичних членів класу.
31. Розвиток навичок роботи з постійними методами.
32. Вдосконалення навичок роботи з віртуальним спадкуванням.
33. Розуміння принципу роботи з чистими віртуальними функціями.
34. Набуття досвіду у використанні множинного спадкування.
35. Розвиток навичок роботи з віртуальними базовими класами.
36. Вдосконалення навичок роботи з шаблонами методів.
37. Розуміння принципу роботи зі спеціалізацією шаблонів.
38. Наявність досвіду у використанні явного інстанціювання шаблонів.
39. Розвиток навичок роботи з параметрами шаблонів для замовчування.
40. Вдосконалення навичок роботи з неявним інстанціюванням шаблонів.
41. Розуміння принципу роботи з варіативними шаблонами.
42. Набуття досвіду у використанні рекурсивних шаблонів.
43. Розвиток навичок роботи з шаблонами змінної тривалості.
44. Вдосконалення навичок роботи з шаблонами з параметрами-пакетами.
45. Розуміння принципу роботи з шаблонами з параметрами-типами.
46. Набуття досвіду у використанні шаблонів з параметрами-значеннями.
47. Розвиток навичок роботи з шаблонами з параметрами-шаблонами.
48. Вдосконалення навичок роботи з шаблонами з параметрами-масивами.
49. Розуміння принципу роботи з шаблонами з параметрами-функціями.
50. Набуття досвіду використання шаблонів з параметрами-методами.
51. Розвиток навичок роботи з шаблонами з параметрами-класами.
52. Вдосконалення навичок роботи з шаблонами з параметрами-об'єктами.
53. Розуміння принципу роботи з шаблонами з параметрами-структурами.
54. Набуття досвіду у використанні шаблонів з параметрами-списками ініціалізації.
55. Розвиток навичок роботи з шаблонами з параметрами-параметрами шаблонів.
56. Вдосконалення навичок роботи з шаблонами з параметрами-параметрами функцій.
57. Розуміння принципу роботи з шаблонами з параметрами-параметрами методів.
58. Набуття досвіду використання шаблонів з параметрами-параметрами класів.
59. Розвиток навичок роботи з шаблонами з параметрами-параметрами об'єктів.
60. Вдосконалення навичок роботи з шаблонами з параметрами-параметрами структури.
61. Розуміння принципу роботи з шаблонами з параметрами-параметрами списку ініціалізації.
62. Набуття досвіду використання шаблонів з параметрами-параметрами шаблонів.
63. Розвиток навичок роботи з шаблонами з параметрами-параметрами функцій.
64. Вдосконалення навичок роботи з шаблонами з параметрами-параметрами методів.
65. Розуміння принципу роботи з шаблонами з параметрами-параметрами класів.
66. Набутий досвід у використанні шаблонів з параметрами-параметрами об'єктів.
67. Розвиток навичок роботи з шаблонами з параметрами-параметрами структури.
68. Вдосконалення навичок роботи з шаблонами з параметрами-параметрами списку ініціалізації.
69. Розуміння принципу роботи з шаблонами з параметрами-параметрами шаблонів.
70. Набуття досвіду використання шаблонів з параметрами-параметрами функцій.
71. Розвиток навичок роботи з шаблонами з параметрами-параметрами методів.
72. Вдосконалення навичок роботи з шаблонами з параметрами-параметрами класів.
73. Розуміння принципу роботи з шаблонами з параметрами-параметрами об'єктів.
74. Набуття досвіду використання шаблонів з параметрами-параметрами структури.
75. Розвиток навичок роботи з шаблонами з параметрами-параметрами списку ініціалізації.

**Відповіді на контрольні запитання:**

**1. Що є результатами виконання концептуалізації предметної області, об'єктного аналізу та визначення інтерфейсів сутностей предметної області?**

Результатами виконання концептуалізації предметної області є визначення ключових сутностей, їх атрибутів та взаємозв'язків між ними. Об'єктний аналіз дозволяє визначити класи та об'єкти, які будуть використовуватися в системі, а також їх поведінку. Визначення інтерфейсів сутностей предметної області включає опис методів та властивостей, які будуть доступні для взаємодії з цими сутностями.

2. **Який зв'язок між процесом концептуалізації предметної області та процесами об'єктного аналізу і визначення інтерфейсів?**

Концептуалізація предметної області є початковим етапом, на якому визначаються основні сутності та їх взаємозв'язки. Об'єктний аналіз базується на цих сутностях і визначає, як вони будуть реалізовані в системі у вигляді об'єктів та класів. Визначення інтерфейсів є завершальним етапом, на якому визначаються методи та властивості, які будуть доступні для взаємодії з цими об'єктами.

**3. Сформулюйте критерії, за якими чітко можливо визначити: абстракцію сутності предметної області слід описати мовою C++ типом структура (struct) чи типом клас (class)?**

Вибір між структурою (struct) та класом (class) залежить від того, як ви плануєте використовувати цю сутність. Якщо сутність має просту структуру даних без складної поведінки та всі члени є публічними, то можна використовувати структуру. Якщо ж сутність має складну поведінку, інкапсуляцію даних та методи для взаємодії з цими даними, то слід використовувати клас.

**4. Що в програмуванні розуміють під інтерфейсом класу?**

Інтерфейс класу — це сукупність публічних методів та властивостей, які визначають, як можна взаємодіяти з об'єктами цього класу. Інтерфейс визначає контракт, який клас повинен виконувати, і дозволяє іншим частинам програми використовувати цей клас без необхідності знати його внутрішню реалізацію.

**5. Обґрунтовано поясніть, чому в класі C++ не можна оголосити конструктор з закритим рівнем доступу?**

Конструктор з закритим рівнем доступу не може бути використаний для створення об'єктів класу поза межами самого класу. Це означає, що ви не зможете створити екземпляри цього класу в інших частинах програми, що робить такий клас непридатним для використання. Конструктори повинні бути публічними, щоб дозволити створення об'єктів класу.

**6. Здійсніть порівняльний аналіз перевантаженої функції та функції з параметрами за замовчуванням.**

Перевантажені функції — це функції з однаковим ім'ям, але з різними параметрами, що дозволяє викликати різні версії функції залежно від передаваних аргументів. Функції з параметрами за замовчуванням дозволяють викликати функцію з меншою кількістю аргументів, ніж оголошено, використовуючи значення за замовчуванням для відсутніх параметрів.

**7. За допомогою яких операторів C++ здійснюється доступ до відкритих членів об'єктів класу?**

Доступ до відкритих членів об'єктів класу здійснюється за допомогою оператора доступу до членів об'єкта . (крапка) для об'єктів та оператора -> для вказівників на об'єкти.

**8. Яким чином клас C++ як абстрактний тип даних (ADT) дозволяє реалізувати принцип інкапсуляції?**

Клас C++ дозволяє реалізувати принцип інкапсуляції, приховуючи внутрішню реалізацію та дані від зовнішнього доступу. Це досягається за допомогою використання різних рівнів доступу (приватний, захищений, публічний), що дозволяє контролювати, які частини класу доступні для зовнішнього використання, а які — ні.