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/](https://github.com/odorenskyi/Dmytro-Parkhomenko-KB18)

ВИКОНАВ

студент академічної групи КІ – 24

Микола Б. С.

ПЕРЕВІРИВ

ст. викладач кафедри кібербезпеки   
та програмного забезпечення

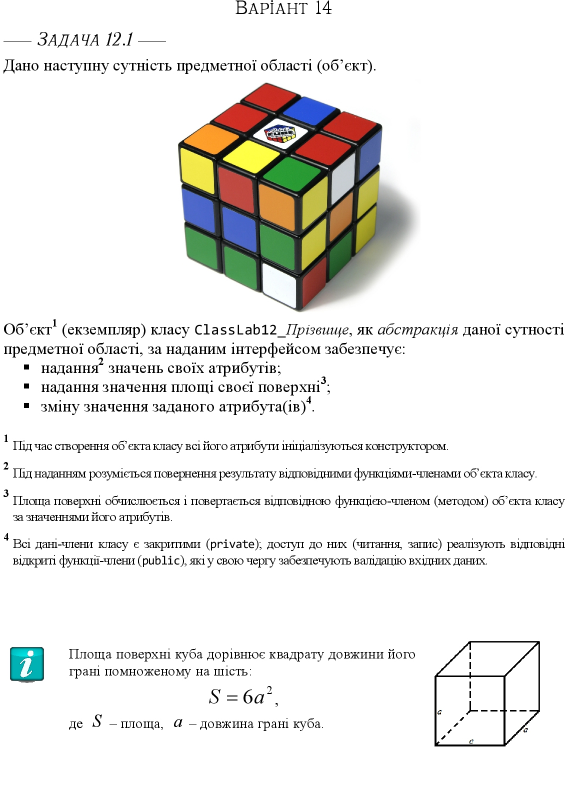
Коваленко А.С

Кропивницький – 2025

**ТЕМА:** Програмна реалізація абстрактних типів даних

**МЕТА** Полягає у набутті ґрунтовних вмінь і практичних навичок об’єктного аналізу й проектування, створення класів С++ та тестування їх екземплярів, використання препроцесорних директив, макросів і макрооператорів під час реалізації програмних засобів у кросплатформовому середовищі Code::Blocks.

**ЗАВДАННЯ №12.1**

****

**ПОСТАНОВКА ЗАДАЧІ:**Розробити програму, яка дозволяє реалізувати абстрактний тип даних для обчислення площі поверхні куба на основі довжини його ребра. Для цього необхідно створити клас, який містить методи для встановлення довжини ребра куба та обчислення площі його поверхні. Програма повинна автоматично генерувати тестові дані, зчитувати їх з файлу, виконувати обчислення та порівнювати результати з очікуваними. Результати тестування мають виводитися у вигляді таблиці на екран та записуватися у вихідний файл.

**АНАЛІЗ ЗАДАЧІ:**Задача передбачає використання принципів об'єктно-орієнтованого програмування для моделювання геометричної фігури — куба. Основним елементом є клас, що реалізує необхідну функціональність згідно з абстрактним типом даних. Обчислення площі поверхні куба виконується за математичною формулою 6 \* a², де a — довжина ребра. Вхідні дані формуються автоматично та зберігаються у текстовому файлі, після чого відбувається зчитування, обробка і порівняння отриманих результатів з контрольними значеннями. Це забезпечує тестування правильності реалізації функцій класу та дає змогу аналізувати точність розрахунків.ч

**РЕАЛІЗАЦІЯ ЗАГОЛОВКОВОГО ФАЙЛУ** ModulesBalan.h

#ifndef MODULESBALAN\_H

#define MODULESBALAN\_H

class ClassLab12\_Cube {

private:

float edge;

public:

ClassLab12\_Cube() : edge(0) {}

void setEdge(float value) { edge = value; }

float getSurfaceArea() {

return 6 \* pow(edge, 2);

}

};

#endif

**РЕАЛІЗАЦІЯ ОСНОВНОЇ ПРОГРАММИ**

#include <iostream>

#include <fstream>

#include <vector>

#include <sstream>

#include <iomanip>

#include <cmath>

#include "ModulesBalan.h"

using namespace std;

struct TestSuite {

float length = 0;

float expSurfaceArea = 0;

float resultSurfaceArea = 0;

};

void createOrUpdateInputFile(const string& filename, const vector<TestSuite>& tests) {

ofstream file(filename);

if (file.is\_open()) {

file << setw(10) << left << "Length" << "|"

<< setw(20) << left << "Expected Surface Area" << "|"

<< setw(20) << left << "Result Surface Area" << "|\n";

file << string(50, '-') << "\n";

for (const auto& test : tests) {

file << fixed << setprecision(2)

<< setw(10) << left << test.length << "|"

<< setw(20) << left << test.expSurfaceArea << "|"

<< setw(20) << left << "" << "|\n";

}

file.close();

}

}

string readOutputFile(const string& filename) {

ifstream file(filename);

string content, line;

if (file.is\_open()) {

while (getline(file, line)) {

if (!line.empty() && line.back() == '\r') {

line.pop\_back();

}

content += line + "\n";

}

file.close();

}

return content;

}

vector<TestSuite> readCVSFiles(const string& filePath) {

ifstream testSuite(filePath);

vector<string> row;

vector<TestSuite> autotest;

TestSuite ts;

string line, value;

if (testSuite.is\_open()) {

getline(testSuite, line);

getline(testSuite, line);

while (getline(testSuite, line)) {

row.clear();

stringstream s(line);

int count = 0;

while (getline(s, value, '|')) {

value.erase(0, value.find\_first\_not\_of(" \t"));

value.erase(value.find\_last\_not\_of(" \t") + 1);

if (!value.empty() && count < 3) {

if (count == 0) ts.length = stof(value);

else if (count == 1) ts.expSurfaceArea = stof(value);

else if (count == 2) ts.resultSurfaceArea = !value.empty() ? stof(value) : 0;

count++;

}

}

if (count >= 2) {

autotest.push\_back(ts);

}

}

testSuite.close();

}

return autotest;

}

void printTable(const vector<TestSuite>& tests) {

cout << setw(10) << left << "Length" << "|"

<< setw(20) << left << "Expected Surface Area" << "|"

<< setw(20) << left << "Result Surface Area" << "|\n";

cout << string(50, '-') << "\n";

for (size\_t i = 0; i < tests.size(); i++) {

cout << fixed << setprecision(2)

<< setw(10) << left << tests[i].length << "|"

<< setw(20) << left << tests[i].expSurfaceArea << "|"

<< setw(20) << left << tests[i].resultSurfaceArea << "|\n";

}

cout << endl;

}

void updateInputFile(const string& filename, const vector<TestSuite>& tests) {

ofstream file(filename);

if (file.is\_open()) {

file << setw(10) << left << "Length" << "|"

<< setw(20) << left << "Expected Surface Area" << "|"

<< setw(20) << left << "Result Surface Area" << "|\n";

file << string(50, '-') << "\n";

for (const auto& test : tests) {

file << fixed << setprecision(2)

<< setw(10) << left << test.length << "|"

<< setw(20) << left << test.expSurfaceArea << "|"

<< setw(20) << left << test.resultSurfaceArea << "|\n";

}

file.close();

}

}

void testSurfaceArea(vector<TestSuite>& tests, ofstream& testResult) {

ClassLab12\_Cube cube;

for (size\_t i = 0; i < tests.size(); i++) {

cube.setEdge(tests[i].length);

tests[i].resultSurfaceArea = cube.getSurfaceArea();

if (fabs(tests[i].resultSurfaceArea - tests[i].expSurfaceArea) < 0.01) {

testResult << "test №" << i + 1 << " (Surface Area) -> passed" << endl << endl;

}

else {

testResult << "test №" << i + 1 << " (Surface Area) -> failed" << endl

<< "answers don't match" << endl

<< "It should be -> " << fixed << setprecision(2) << tests[i].expSurfaceArea << " sq cm" << endl

<< "Received response -> " << fixed << setprecision(2) << tests[i].resultSurfaceArea << " sq cm" << endl << endl;

}

}

}

int main() {

const string inputFilePath = "TS\_Cube.txt";

vector<TestSuite> defaultTests = {

{2.0, 24.00},

{3.0, 54.00},

{4.0, 96.00},

{5.0, 150.00},

{6.0, 216.00}

};

createOrUpdateInputFile(inputFilePath, defaultTests);

ofstream testResult("TestResult\_Cube.txt");

if (!testResult.is\_open()) {

cout << "Не вдалося створити файл результатів!" << endl;

return -1;

}

vector<TestSuite> autotest = readCVSFiles(inputFilePath);

if (!autotest.empty()) {

testSurfaceArea(autotest, testResult);

printTable(autotest);

}

else {

cout << "Помилка: файл TS\_Cube.txt порожній або не знайдено!" << endl;

}

if (!autotest.empty()) {

updateInputFile(inputFilePath, autotest);

}

testResult.close();

string outputContent = readOutputFile("TestResult\_Cube.txt");

cout << outputContent;

system("pause");

return 0;

}

**РЕЗУЛЬТАТ ВИКОНАННЯ ПРОГРАММИ:**

Length |Expected Surface Area|Result Surface Area |

--------------------------------------------------

2.00 |24.00 |24.00 |

3.00 |54.00 |54.00 |

4.00 |96.00 |96.00 |

5.00 |150.00 |150.00 |

6.00 |216.00 |216.00 |

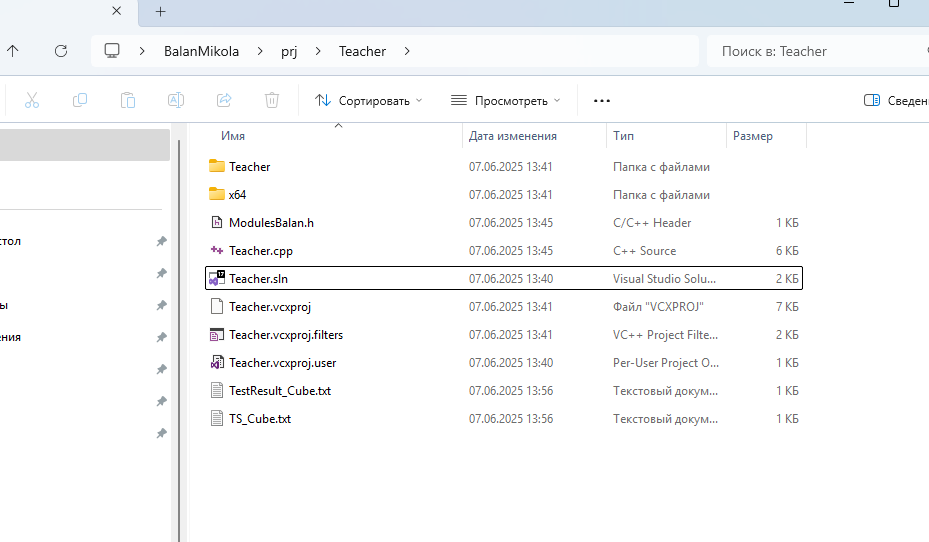
test ╣1 (Surface Area) -> passed

test ╣2 (Surface Area) -> passed

test ╣3 (Surface Area) -> passed

test ╣4 (Surface Area) -> passed

test ╣5 (Surface Area) -> passed



Малюнок 1 - Файли програми після запуску

**ВІДПОВІДЬ НА КОНТРОЛЬНІ ЗАПИТАННЯ**

1. **Що є результатами виконання концептуалізації предметної області, об’єктного аналізу та визначення інтерфейсів сутностей предметної області?**  
   У результаті цих етапів визначаються ключові об’єкти предметної області, їхні характеристики та взаємозв’язки, формується структура класів або структур у коді, а також описуються інтерфейси для взаємодії між об’єктами системи.
2. **Який зв’язок між процесом концептуалізації предметної області та процесами об’єктного аналізу і визначення інтерфейсів?**  
   Концептуалізація задає основу, на якій ґрунтується об’єктний аналіз — вона визначає сутності та їхні відносини. Об’єктний аналіз деталізує ці сутності у вигляді об’єктів з атрибутами та методами. Визначення інтерфейсів забезпечує взаємодію цих об’єктів у програмному середовищі.
3. **Сформулюйте критерії, за якими чітко можливо визначити: абстракцію сутності предметної області слід описати мовою С++ типом структура (struct) чи типом клас (class)?**  
   Struct використовують, коли потрібно зберігати лише набір даних без логіки. Class доцільно застосовувати, коли сутність має як дані, так і методи, або коли необхідно реалізувати інкапсуляцію та контроль доступу.
4. **Що в програмуванні розуміють під інтерфейсом класу?**  
   Інтерфейс класу — це сукупність публічних методів і властивостей, через які користувач взаємодіє з об’єктом, не знаючи його внутрішньої реалізації.
5. **Обґрунтовано поясніть, чому в класі С++ не можна оголосити конструктор з закритим рівнем доступу?**  
   Якщо конструктор приватний, створення об’єкта поза межами класу стане неможливим. Це дозволено лише в особливих випадках, наприклад, у шаблоні Singleton. У загальному випадку клас з лише приватним конструктором обмежує своє використання.
6. **Здійсніть порівняльний аналіз перевантаженої функції та функції з параметрами за замовчуванням.**  
   Перевантаження — це створення кількох варіантів функції з однаковою назвою, але різними параметрами. Функція з параметрами за замовчуванням — це одна функція, де деякі аргументи можуть не передаватися. Перевантаження забезпечує більшу гнучкість, а параметри за замовчуванням — компактність коду.
7. **За допомогою яких операторів С++ здійснюється доступ до відкритих членів об’єктів класу?**  
   Використовують оператор . для об’єктів та -> для вказівників на об’єкти.
8. **Яким чином клас С++ як абстрактний тип даних (ADT) дозволяє реалізувати принцип інкапсуляції?**  
   Клас дозволяє приховати внутрішню реалізацію за допомогою рівнів доступу (private, protected) і відкривати лише потрібний функціонал через public інтерфейс. Це дозволяє контролювати доступ до даних і зберігати цілісність об’єкта.

**ВИСНОВКИ**

1. Засвоєно поняття абстрактного типу даних (АТД).
2. Розглянуто реалізацію АТД мовою C++.
3. Набуто навичок створення класів у C++.
4. Здійснено об’єктний аналіз предметної області.
5. Реалізовано структуру даних через класи.
6. Практично використано принцип інкапсуляції.
7. Освоєно механізм доступу до членів класу.
8. Використано модифікатори доступу.
9. Створено публічний інтерфейс класу.
10. Вивчено принципи проектування класів.
11. Освоєно основи ООП у C++.
12. Реалізовано зв’язки між об’єктами.
13. Впроваджено методи в класах.
14. Освоєно створення конструкторів.
15. Навченося створювати деструктори.
16. Вивчено перевантаження методів.
17. Розглянуто використання шаблонів.
18. Реалізовано класи з динамічною пам’яттю.
19. Освоєно ініціалізацію об’єктів.
20. Засвоєно роботу з вказівниками на об’єкти.
21. Використано препроцесорні директиви.
22. Навченося використовувати макроси.
23. Вивчено макрооператори # та ##.
24. Реалізовано інкапсуляцію даних.
25. Використано директиву #define.
26. Застосовано умови компіляції #ifdef.
27. Створено кросплатформний код.
28. Здійснено тестування класів.
29. Розроблено інтерфейс класу.
30. Освоєно поняття приватних і публічних методів.
31. Навченося організовувати заголовочні файли.
32. Використано директиву #include.
33. Розглянуто ізоляцію реалізації через .cpp і .h файли.
34. Створено кілька об’єктів одного класу.
35. Освоєно використання масивів об’єктів.
36. Здійснено перегрузку операторів.
37. Практикувався контроль доступу.
38. Використано ініціалізаційні списки.
39. Вивчено створення компіляційних умов.
40. Практично реалізовано інтерфейси взаємодії.
41. Застосовано принципи поліморфізму.
42. Використано абстрактні класи.
43. Освоєно створення віртуальних функцій.
44. Здійснено модульне програмування.
45. Вивчено структуру проекту в Code::Blocks.
46. Практично застосовано середовище Code::Blocks.
47. Реалізовано функціональність об’єктів.
48. Створено функції-члени класів.
49. Використано оператор this.
50. Навченося створювати об’єкти динамічно.
51. Застосовано new та delete.
52. Проведено аналіз потреб у класах.
53. Розроблено модель даних у вигляді класів.
54. Вивчено механізм конструкторів за замовчуванням.
55. Реалізовано перегрузку конструкторів.
56. Визначено різницю між struct і class.
57. Освоєно створення приватних змінних.
58. Забезпечено контроль доступу до даних.
59. Побудовано логіку взаємодії між класами.
60. Використано функції з параметрами за замовчуванням.
61. Освоєно механізм агрегації об’єктів.
62. Реалізовано дружні функції.
63. Створено приклади використання методів.
64. Визначено принципи проектування об’єктів.
65. Застосовано шаблони для розширення функцій.
66. Визначено ключові елементи АТД.
67. Проведено аналіз задачі на основі АТД.
68. Узагальнено знання про ООП.
69. Використано множинне включення #pragma once.
70. Забезпечено повторне використання коду.
71. Проведено відлагодження проєкту.
72. Вивчено принцип спадкування.
73. Реалізовано базові та похідні класи.
74. Розглянуто реалізацію списків та черг.
75. Освоєно роботу з вектором об’єктів.
76. Реалізовано вкладені класи.
77. Проведено тестування на коректність виконання.
78. Навченося створювати зручні інтерфейси.
79. Застосовано гнучке проектування.
80. Освоєно моделювання поведінки об’єктів.
81. Створено класи з логікою обробки.
82. Проведено аналіз помилок компіляції.
83. Узагальнено структуру проєкту.
84. Набуто навичок кодування за правилами ООП.
85. Опрацьовано механізми захисту даних.
86. Розроблено класи з повною функціональністю.
87. Реалізовано базову архітектуру проєкту.
88. Побудовано логіку через методи класів.
89. Вивчено роботу зі статичними членами.
90. Застосовано змінні та методи класу.
91. Розширено розуміння ООП-конструкцій.
92. Узгоджено специфікацію методів класу.
93. Побудовано класову модель системи.
94. Опрацьовано принцип єдності інтерфейсу.
95. Використано типи даних у класах.
96. Проведено розділення оголошень і реалізацій.
97. Освоєно обробку вхідних/вихідних даних у класах.
98. Визначено ролі методів у класі.
99. Розглянуто життєвий цикл об’єкта.
100. Узагальнено навички програмної реалізації АТД.