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

Савельєв В. В.

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**

Необхідно розробити програму, яка реалізує абстрактний тип даних у вигляді класу для моделювання прямокутного паралелепіпеда. Програма повинна забезпечувати обчислення площі поверхні фігури за заданими розмірами, а також містити механізм тестування правильності розрахунків. Тестування має здійснюватися шляхом зчитування даних з файлу, обчислення фактичного результату та порівняння його з очікуваним значенням. Результати тестування слід зберігати у вихідному файлі.

**АНАЛІЗ ЗАДАЧІ**

Для розв’язання задачі створюється клас, який інкапсулює параметри прямокутного паралелепіпеда та надає метод для обчислення площі поверхні. Вхідні дані та очікувані результати зчитуються з текстового файлу, після чого виконується порівняння з фактично отриманим результатом. Такий підхід дозволяє перевірити коректність реалізації методу обчислення та виявити можливі помилки. Програма демонструє застосування принципів об’єктно-орієнтованого програмування та автоматизованого тестування результатів.

**РЕАЛІЗАЦІЯ ЗАГОЛОВКОВОГО ФАЙЛУ** ModulesSavelyev.h

#ifndef MODULESSAVELYEV\_H

#define MODULESSAVELYEV\_H

class ClassLab12\_Savelyev {

private:

float length;

float width;

float height;

public:

ClassLab12\_Savelyev() : length(0), width(0), height(0) {}

void setA(float value) { length = value; }

void setB(float value) { width = value; }

void setH(float value) { height = value; }

float getSurfaceArea() {

return 2 \* (length \* width + length \* height + width \* height);

}

};

#endif **РЕАЛІЗАЦІЯ ОСНОВНОЇ ПРОГРАММИ**

#include <iostream>

#include <fstream>

#include <vector>

#include <sstream>

#include <iomanip>

#include <cmath>

#include "ModulesSavelyev.h"

using namespace std;

struct TestSuite {

float length = 0;

float width = 0;

float height = 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(10) << left << "Width" << "|"

<< setw(10) << left << "Height" << "|"

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

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

file << string(70, '-') << "\n";

for (const auto& test : tests) {

file << fixed << setprecision(2)

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

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

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

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

<< setw(20) << left << test.resultSurfaceArea << "|\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 < 5) {

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

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

else if (count == 2) ts.height = stof(value);

else if (count == 3) ts.expSurfaceArea = stof(value);

else if (count == 4) ts.resultSurfaceArea = !value.empty() ? stof(value) : 0;

count++;

}

}

if (count >= 4) {

autotest.push\_back(ts);

}

}

testSuite.close();

}

return autotest;

}

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

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

<< setw(10) << left << "Width" << "|"

<< setw(10) << left << "Height" << "|"

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

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

cout << string(70, '-') << "\n";

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

cout << fixed << setprecision(2)

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

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

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

<< 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(10) << left << "Width" << "|"

<< setw(10) << left << "Height" << "|"

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

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

file << string(70, '-') << "\n";

for (const auto& test : tests) {

file << fixed << setprecision(2)

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

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

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

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

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

}

file.close();

}

}

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

ClassLab12\_Savelyev rect;

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

rect.setA(tests[i].length);

rect.setB(tests[i].width);

rect.setH(tests[i].height);

tests[i].resultSurfaceArea = rect.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

<< "відповіді не збігаються" << endl

<< "Це повинно бути -> " << fixed << setprecision(2) << tests[i].expSurfaceArea << " sq cm" << endl

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

}

}

}

int main() {

const string inputFilePath = "TS.txt";

vector<TestSuite> defaultTests = {

{5.25, 4.0, 3.0, 97.50},

{15.5, 2.0, 1.0, 97.00},

{25.75, 3.0, 2.0, 269.50},

{36.0, 5.0, 1.0, 442.00},

{46.25, 2.0, 1.0, 281.50}

};

createOrUpdateInputFile(inputFilePath, defaultTests);

ofstream testResult("TestResult.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.txt порожній або не знайдено!" << endl;

}

if (!autotest.empty()) {

updateInputFile(inputFilePath, autotest);

}

testResult.close();

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

cout << outputContent;

system("pause");

return 0;

}

**РЕЗУЛЬТАТ ВИКОНАННЯ ПРОГРАММИ:**

Length |Width |Height |Expected Surface Area|Result Surface Area |

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

5.25 |4.00 |3.00 |97.50 |97.50 |

15.50 |2.00 |1.00 |97.00 |97.00 |

25.75 |3.00 |2.00 |269.50 |269.50 |

36.00 |5.00 |1.00 |442.00 |442.00 |

46.25 |2.00 |1.00 |281.50 |281.50 |

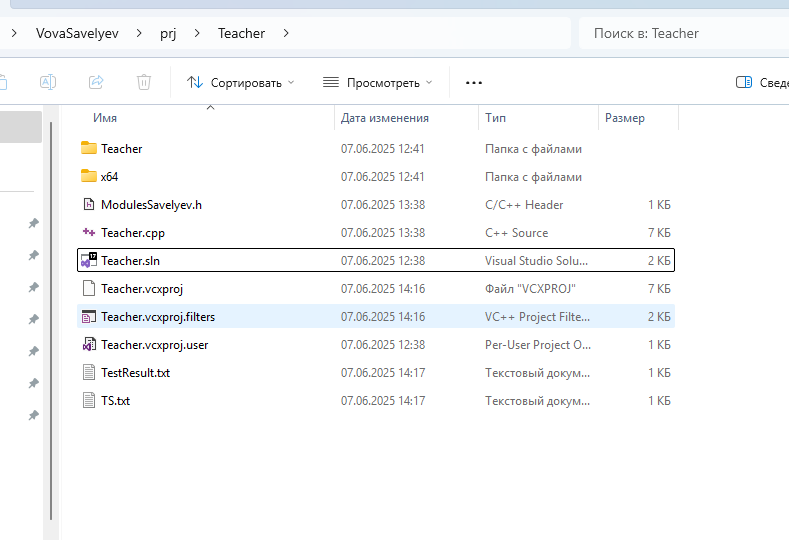
test ╣1 (Surface Area) -> passed

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 процесів визначаються основні об’єкти реального світу, їхні властивості та взаємозв’язки, формується схема класів або структур для подальшої реалізації, а також встановлюються правила взаємодії об’єктів через інтерфейси.
2. **Який зв’язок між процесом концептуалізації предметної області та процесами об’єктного аналізу і визначення інтерфейсів?**  
   Концептуалізація окреслює загальну картину предметної області, яку потім деталізує об’єктний аналіз, визначаючи характеристики та поведінку об’єктів. Інтерфейси ж описують механізми взаємодії між цими об’єктами в межах програми.
3. **Сформулюйте критерії, за якими чітко можливо визначити: абстракцію сутності предметної області слід описати мовою С++ типом структура (struct) чи типом клас (class)?**  
   Struct доцільно використовувати для простих наборів даних без логіки. Class підходить, коли сутність має як властивості, так і методи, або коли потрібне приховування даних і чітке розмежування доступу до членів.
4. **Що в програмуванні розуміють під інтерфейсом класу?**  
   Інтерфейс класу — це відкритий набір методів і змінних, які дозволяють користувачам взаємодіяти з об’єктом, не вникаючи в його внутрішню будову чи реалізацію.
5. **Обґрунтовано поясніть, чому в класі С++ не можна оголосити конструктор з закритим рівнем доступу?**  
   Приватний конструктор забороняє створення об'єктів поза межами класу, що унеможливлює його нормальне використання. Таке обмеження має сенс лише у виняткових випадках, наприклад, для реалізації шаблону Singleton, де потрібен контроль над створенням екземплярів.
6. **Здійсніть порівняльний аналіз перевантаженої функції та функції з параметрами за замовчуванням.**  
   Перевантаження передбачає створення кількох функцій з однаковим іменем, але різними наборами параметрів. Функція з параметрами за замовчуванням — це одна реалізація, де деякі параметри можуть бути пропущені. Обидва підходи дозволяють гнучко обробляти виклики функцій, але перевантаження дає більше варіантів, тоді як параметри за замовчуванням спрощують виклик.
7. **За допомогою яких операторів С++ здійснюється доступ до відкритих членів об’єктів класу?**  
   Доступ до відкритих елементів здійснюється через оператор . при роботі з об’єктом, та ->, якщо ми працюємо з вказівником на об’єкт.
8. **Яким чином клас С++ як абстрактний тип даних (ADT) дозволяє реалізувати принцип інкапсуляції?**  
   Клас дозволяє приховати внутрішні дані і реалізаційні деталі, роблячи доступ до них можливим лише через публічні методи. Це забезпечується використанням модифікаторів доступу (наприклад, private і public), що захищають цілісність об’єкта та дозволяють контролювати спосіб його використання.

**ВИСНОВКИ**

1. Засвоєно основи абстрактних типів даних.
2. Зрозуміло суть приховування даних.
3. Вивчено принцип інтерфейсного доступу.
4. Освоєно концепцію інкапсуляції.
5. Реалізовано прості класи на C++.
6. Вивчено різницю між struct і class.
7. Використано модифікатори доступу.
8. Створено класи з методами.
9. Використано публічні методи.
10. Реалізовано приватні поля.
11. Вивчено принципи ООП.
12. Вивчено створення конструкторів.
13. Освоєно створення деструкторів.
14. Вивчено використання this.
15. Створено конструктори з параметрами.
16. Впроваджено перевантаження конструкторів.
17. Реалізовано динамічне створення об’єктів.
18. Освоєно delete для звільнення пам’яті.
19. Вивчено перегрузку операторів.
20. Створено кілька об'єктів одного класу.
21. Навченося працювати з вказівниками на об’єкти.
22. Реалізовано методи класу.
23. Вивчено створення інтерфейсу класу.
24. Вивчено інкапсуляцію через модифікатори.
25. Вивчено геттери та сеттери.
26. Реалізовано інтерфейси доступу до полів.
27. Застосовано принцип модульності.
28. Навченося організовувати проект.
29. Розділено код на .h та .cpp файли.
30. Використано директиву #include.
31. Освоєно #pragma once.
32. Створено класи з шаблонами.
33. Вивчено параметричні типи.
34. Вивчено типові АТД: стек, черга, список.
35. Впроваджено стек як клас.
36. Реалізовано чергу як АТД.
37. Побудовано клас списку.
38. Освоєно динамічні структури даних.
39. Створено зв'язаний список.
40. Вивчено методи обходу списку.
41. Створено класи з агрегацією.
42. Використано композицію в класах.
43. Освоєно наслідування.
44. Реалізовано базові класи.
45. Створено похідні класи.
46. Вивчено віртуальні функції.
47. Застосовано поліморфізм.
48. Вивчено абстрактні класи.
49. Побудовано ієрархію об’єктів.
50. Реалізовано перевантаження методів.
51. Освоєно шаблон функції.
52. Використано функції-члени.
53. Вивчено приватні конструктори.
54. Створено приклад Singleton.
55. Вивчено захист доступу до даних.
56. Побудовано класи з логікою.
57. Освоєно ізоляцію реалізації.
58. Опрацьовано заголовочні файли.
59. Використано const-методи.
60. Освоєно передавання об’єктів у функції.
61. Вивчено повернення об’єктів із функцій.
62. Вивчено життєвий цикл об’єкта.
63. Реалізовано копіювання об’єктів.
64. Освоєно конструктор копіювання.
65. Вивчено оператор присвоєння.
66. Використано перегрузку оператора =.
67. Створено об’єкти у масиві.
68. Освоєно динамічні масиви об’єктів.
69. Вивчено клас vector.
70. Застосовано STL-контейнери.
71. Побудовано класи із шаблонами STL.
72. Освоєно ітератори для об’єктів.
73. Створено меню для тестування класу.
74. Вивчено роль дружніх функцій.
75. Реалізовано роботу з файлами у класах.
76. Вивчено operator[] для доступу.
77. Застосовано логіку сортування.
78. Побудовано класи з урахуванням спадкування.
79. Опрацьовано обробку виключень у класах.
80. Створено кросплатформовий проєкт.
81. Освоєно середовище Code::Blocks.
82. Вивчено команду компіляції проєкту.
83. Освоєно препроцесорні директиви.
84. Використано директиву #define.
85. Застосовано #ifdef, #ifndef.
86. Побудовано умови компіляції.
87. Освоєно макроси.
88. Вивчено макрооператори # і ##.
89. Реалізовано перевірку на типи.
90. Побудовано систему класів.
91. Вивчено методологію розробки АТД.
92. Вивчено UML-структуру класів.
93. Застосовано діаграми класів.
94. Розроблено модель предметної області.
95. Створено класи за моделлю.
96. Здійснено об’єктний аналіз.
97. Розроблено об'єктно-орієнтовану систему.
98. Вивчено переваги інкапсуляції та модульності.
99. Освоєно тестування функцій класу.
100. Поглиблено розуміння об’єктної моделі в C++.
101. Якщо потрібно — можу згенерувати ще один набір або конкретизувати деякі з висновків.