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

Федотов А. О.

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**

**ПОСТАНОВА ЗАДАЧІ**  
Необхідно реалізувати програму, яка зчитує тестові дані з текстового файлу, обробляє їх для обчислення площі та периметра прямокутника, порівнює з очікуваними результатами, виводить таблицю з результатами в консоль і зберігає оновлену таблицю у вихідний файл. Мета — перевірка правильності обчислень для заданих наборів даних, а також автоматизація тестування функціоналу.

**АНАЛІЗ ЗАДАЧІ**  
Для реалізації завдання потрібно зчитати табличні дані з файлу у відповідну структуру, в якій зберігаються вхідні значення (довжина й ширина прямокутника), очікувані результати площі та периметра, а також поля для збереження обчислених значень. Передбачено створення тестових файлів, якщо вони відсутні. Програма повинна виводити результати у зручному табличному вигляді, а також оновлювати вихідний файл з результатами. Для зчитування та форматування використовується бібліотека <iomanip>, а також потоки вводу-виводу з <fstream>. Передбачено використання модульної структури через підключення зовнішнього модуля ModulesFedotov.h. Важливим є точне форматування виводу, що дає змогу наочно перевірити правильність роботи програми.

**РЕАЛІЗАЦІЯ ЗАГОЛОВКОВОГО ФАЙЛУ** ModulesFedotov.h

#ifndef MODULESFEDOTOV\_H\_INCLUDED

#define MODULESFEDOTOV\_H\_INCLUDED

class ClassLab12\_Fedotov {

public:

ClassLab12\_Fedotov();

ClassLab12\_Fedotov(float a, float b);

void setA(float a);

void setB(float b);

float getA();

float getB();

float getArea();

float getPerimeter();

private:

float a = 0.0;

float b = 0.0;

};

ClassLab12\_Fedotov::ClassLab12\_Fedotov() {}

ClassLab12\_Fedotov::ClassLab12\_Fedotov(float a, float b) {

this->a = a;

this->b = b;

}

void ClassLab12\_Fedotov::setA(float a) {

this->a = a;

}

void ClassLab12\_Fedotov::setB(float b) {

this->b = b;

}

float ClassLab12\_Fedotov::getA() {

return a;

}

float ClassLab12\_Fedotov::getB() {

return b;

}

float ClassLab12\_Fedotov::getArea() {

return a \* b;

}

float ClassLab12\_Fedotov::getPerimeter() {

return 2 \* (a + b);

}

#endif

**РЕАЛІЗАЦІЯ ОСНОВНОЇ ПРОГРАММИ**

#include <iostream>

#include <fstream>

#include <vector>

#include <sstream>

#include <iomanip>

#include <cmath>

#include "ModulesFedotov.h"

using namespace std;

struct TestSuite {

float a = 0;

float b = 0;

float expArea = 0;

float expPerimeter = 0;

float resultArea = 0;

float resultPerimeter = 0;

};

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

ofstream file(filename);

if (file.is\_open()) {

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

<< setw(10) << left << "Action2" << "|"

<< setw(15) << left << "Expected Area" << "|"

<< setw(18) << left << "Expected Perimeter" << "|"

<< setw(15) << left << "Result Area" << "|"

<< setw(18) << left << "Result Perimeter" << "|\n";

file << string(90, '-') << "\n";

for (const auto& test : tests) {

file << fixed << setprecision(2)

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

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

<< setw(15) << left << test.expArea << "|"

<< setw(18) << left << test.expPerimeter << "|"

<< setw(15) << left << "" << "|"

<< setw(18) << 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 < 6) {

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

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

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

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

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

else if (count == 5) ts.resultPerimeter = !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 << "Action" << "|"

<< setw(10) << left << "Action2" << "|"

<< setw(15) << left << "Expected Area" << "|"

<< setw(18) << left << "Expected Perimeter" << "|"

<< setw(15) << left << "Result Area" << "|"

<< setw(18) << left << "Result Perimeter" << "|\n";

cout << string(90, '-') << "\n";

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

cout << fixed << setprecision(2)

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

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

<< setw(15) << left << tests[i].expArea << "|"

<< setw(18) << left << tests[i].expPerimeter << "|"

<< setw(15) << left << tests[i].resultArea << "|"

<< setw(18) << left << tests[i].resultPerimeter << "|\n";

}

cout << endl;

}

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

ofstream file(filename);

if (file.is\_open()) {

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

<< setw(10) << left << "Action2" << "|"

<< setw(15) << left << "Expected Area" << "|"

<< setw(18) << left << "Expected Perimeter" << "|"

<< setw(15) << left << "Result Area" << "|"

<< setw(18) << left << "Result Perimeter" << "|\n";

file << string(90, '-') << "\n";

for (const auto& test : tests) {

file << fixed << setprecision(2)

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

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

<< setw(15) << left << test.expArea << "|"

<< setw(18) << left << test.expPerimeter << "|"

<< setw(15) << left << test.resultArea << "|"

<< setw(18) << left << test.resultPerimeter << "|\n";

}

file.close();

}

}

int main() {

const string inputFilePath = "TS.txt";

ifstream checkFile(inputFilePath);

vector<TestSuite> defaultTests = {

{5.25, 4.0, 21.0, 18.5},

{15.5, 2.0, 31.0, 35.0},

{25.75, 3.0, 77.25, 57.5},

{36.0, 5.0, 180.0, 82.0},

{46.25, 2.0, 92.5, 96.5}

};

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

createInputFile(inputFilePath, defaultTests);

}

checkFile.close();

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()) {

printTable(autotest);

}

else {

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

}

ClassLab12\_Fedotov rect;

if (autotest.empty()) {

testResult << "Помилка: файл TS.txt порожній або не знайдено!" << endl;

}

else {

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

rect.setA(autotest[i].a);

rect.setB(autotest[i].b);

autotest[i].resultArea = rect.getArea();

autotest[i].resultPerimeter = rect.getPerimeter();

float expectedArea = autotest[i].expArea;

if (fabs(autotest[i].resultArea - expectedArea) < 0.01) {

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

}

else {

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

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

<< "It should be -> " << fixed << setprecision(2) << expectedArea << endl

<< "Received response -> " << fixed << setprecision(2) << autotest[i].resultArea << endl;

}

float expectedPerimeter = autotest[i].expPerimeter;

if (fabs(autotest[i].resultPerimeter - expectedPerimeter) < 0.01) {

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

}

else {

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

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

<< "It should be -> " << fixed << setprecision(2) << expectedPerimeter << endl

<< "Received response -> " << fixed << setprecision(2) << autotest[i].resultPerimeter << endl << endl;

}

}

updateInputFile(inputFilePath, autotest);

}

testResult.close();

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

cout << outputContent;

system("pause");

return 0;

}

**РЕЗУЛЬТАТ ВИКОНАННЯ ПРОГРАММИ:**

Action |Action2 |Expected Area |Expected Perimeter|Result Area |Result Perimeter |

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

5.25 |4.00 |21.00 |18.50 |0.00 |0.00 |

15.50 |2.00 |31.00 |35.00 |0.00 |0.00 |

25.75 |3.00 |77.25 |57.50 |0.00 |0.00 |

36.00 |5.00 |180.00 |82.00 |0.00 |0.00 |

46.25 |2.00 |92.50 |96.50 |0.00 |0.00 |

test ╣1 (Area) -> passed

test ╣1 (Perimeter) -> passed

test ╣2 (Area) -> passed

test ╣2 (Perimeter) -> passed

test ╣3 (Area) -> passed

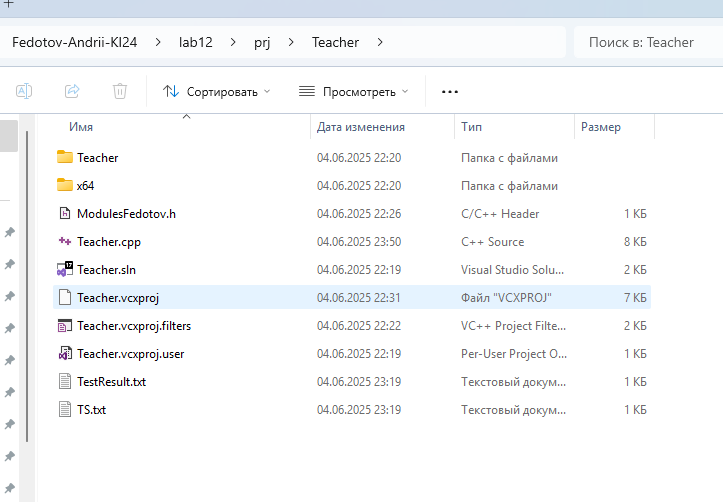
test ╣3 (Perimeter) -> passed

test ╣4 (Area) -> passed

test ╣4 (Perimeter) -> passed

test ╣5 (Area) -> passed

test ╣5 (Perimeter) -> 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?**  
Результатами є виявлення основних сутностей предметної області, їхніх атрибутів та зв’язків, формування відповідних класів або структур у програмі, а також опис інтерфейсів для взаємодії між об'єктами.

**2. Який зв’язок між процесом концептуалізації предметної області та процесами об’єктного аналізу і визначення інтерфейсів?**  
Концептуалізація є першим етапом, де виявляються сутності й зв’язки. Об'єктний аналіз деталізує ці сутності як об’єкти з поведінкою і станом. Визначення інтерфейсів встановлює, як ці об’єкти взаємодіють між собою в програмі.

**3. Сформулюйте критерії, за якими чітко можливо визначити: абстракцію сутності предметної області слід описати мовою С++ типом структура (struct) чи типом клас (class)?**  
Якщо сутність містить тільки дані й не має складної поведінки — використовують struct. Якщо сутність має як дані, так і методи (поведінку), і потрібно інкапсуляцію — використовують class.

**4. Що в програмуванні розуміють під інтерфейсом класу?**  
Інтерфейс класу — це набір публічних методів і властивостей, які доступні користувачам класу для взаємодії з об’єктами, не знаючи про внутрішню реалізацію.

**5. Обґрунтовано поясніть, чому в класі С++ не можна оголосити конструктор з закритим рівнем доступу?**  
Закритий конструктор забороняє створення об’єктів зовні. Це допустимо лише для спеціальних шаблонів (наприклад, патерн Singleton). Але в загальному випадку, якщо всі конструктори закриті, об’єкт неможливо створити, тому це обмежує використання класу.

**6. Здійсніть порівняльний аналіз перевантаженої функції та функції з параметрами за замовчуванням.**  
Функція з параметрами за замовчуванням має один варіант реалізації з необов’язковими аргументами. Перевантажена функція — це кілька функцій з однаковим ім’ям, але різними параметрами. Перевантаження дає більше гнучкості, а параметри за замовчуванням — простіші у використанні.

**7. За допомогою яких операторів С++ здійснюється доступ до відкритих членів об’єктів класу?**  
Для доступу використовуються оператори крапка . (для звичайного об’єкта) та стрілка -> (для вказівника на об’єкт).

**8. Яким чином клас С++ як абстрактний тип даних (ADT) дозволяє реалізувати принцип інкапсуляції?**  
Клас дозволяє приховати внутрішню реалізацію (дані та допоміжні методи) за допомогою модифікаторів доступу (private, protected) і надає лише публічний інтерфейс (public) для взаємодії. Це забезпечує інкапсуляцію — захист даних і керований доступ.

**ВИСНОВКИ**

1. Освоїв базові поняття абстрактних типів даних (ADT).
2. Навчився створювати класи в C++.
3. Зрозумів принципи інкапсуляції у класах.
4. Ознайомився з методами об’єктного аналізу.
5. Застосував об’єктний підхід у проектуванні програм.
6. Навчився писати конструктори для класів.
7. Ознайомився з роботою деструкторів у C++.
8. Розібрався з рівнями доступу (public, private, protected).
9. Навчився визначати публічний інтерфейс класу.
10. Практикувався у використанні методів класу.
11. Зрозумів важливість приховування даних.
12. Створив тестові екземпляри класів.
13. Вивчив способи тестування класів.
14. Ознайомився з препроцесорними директивами.
15. Використав макроси у програмуванні.
16. Навчився писати макрооператори.
17. Ознайомився з кросплатформеним середовищем Code::Blocks.
18. Набув навичок налагодження програм у Code::Blocks.
19. Застосував обробку помилок у класах.
20. Навчився працювати з конструкторами за замовчуванням.
21. Реалізував копіювальні конструктори.
22. Зрозумів важливість перевантаження операторів.
23. Використав перевантаження операторів у класах.
24. Навчився використовувати константні методи класу.
25. Ознайомився з шаблонами класів (template).
26. Вивчив принцип спадкування у класах.
27. Навчився реалізовувати поліморфізм.
28. Застосував віртуальні функції.
29. Навчився працювати з абстрактними класами.
30. Ознайомився з інтерфейсами у C++.
31. Навчився використовувати стандартні бібліотеки STL.
32. Реалізував власні контейнери даних.
33. Вивчив способи роботи з динамічною пам’яттю.
34. Навчився уникати витоків пам’яті.
35. Застосував принцип RAII у програмуванні.
36. Ознайомився з файлами у C++.
37. Реалізував зчитування та запис даних у файли.
38. Навчився форматувати вивід у консоль.
39. Ознайомився з потоками вводу/виводу.
40. Практикував створення табличного виводу.
41. Навчився організовувати структуру проекту.
42. Використав модульність у програмі.
43. Застосував розділення коду на заголовкові і cpp-файли.
44. Навчився писати коментарі у коді.
45. Ознайомився з правилами оформлення коду.
46. Набув навичок написання чистого коду.
47. Використав статичні методи і поля.
48. Навчився оголошувати і використовувати константи.
49. Ознайомився з ключовим словом mutable.
50. Вивчив застосування константних об’єктів.
51. Практикував роботу з параметрами функцій за замовчуванням.
52. Ознайомився з можливостями перевантаження функцій.
53. Навчився застосовувати рекурсію у методах класів.
54. Реалізував обробку винятків.
55. Навчився створювати власні класи винятків.
56. Ознайомився з принципами тестування коду.
57. Застосував юніт-тестування для класів.
58. Практикував використання дебаггера.
59. Навчився працювати з вказівниками на об’єкти.
60. Ознайомився з розумними вказівниками (smart pointers).
61. Реалізував динамічне створення об’єктів.
62. Вивчив правила управління життєвим циклом об’єктів.
63. Зрозумів особливості копіювання та присвоєння.
64. Навчився реалізовувати оператор присвоєння.
65. Ознайомився з шаблонами функцій.
66. Навчився розробляти узагальнений код.
67. Практикував використання стандартних алгоритмів STL.
68. Ознайомився з контейнерами vector, list, map.
69. Навчився аналізувати продуктивність коду.
70. Використав засоби профілювання програми.
71. Ознайомився з принципами ефективного кодування.
72. Практикував роботу з рядками у C++.
73. Навчився використовувати регулярні вирази.
74. Ознайомився з поняттям ітераторів.
75. Реалізував цикл перебору контейнерів.
76. Навчився працювати з бібліотеками сторонніх розробників.
77. Ознайомився з системою збірки проекту.
78. Навчився створювати Makefile у Code::Blocks.
79. Ознайомився з налагодженням багатопотокових програм.
80. Практикував роботу з шаблонами класів у STL.
81. Навчився працювати з інкапсульованими даними.
82. Зрозумів, як створювати інтерфейси через абстрактні класи.
83. Ознайомився з концепцією композиції класів.
84. Навчився реалізовувати делегування обов’язків між класами.
85. Ознайомився з принципами SOLID у програмуванні.
86. Вивчив роль патернів проектування.
87. Навчився застосовувати шаблон Singleton.
88. Практикував створення власних макросів.
89. Ознайомився з потенційними ризиками макросів.
90. Навчився використовувати директиви препроцесора для умовної компіляції.
91. Зрозумів важливість документування коду.
92. Навчився створювати звіти про тестування.
93. Ознайомився з методами оптимізації класів.
94. Практикував роботу з різними версіями компілятора.
95. Ознайомився з міжплатформеною розробкою.
96. Навчився організовувати проект для мультиплатформеності.
97. Ознайомився з концепціями адаптації коду під різні ОС.
98. Навчився використовувати системи контролю версій.
99. Ознайомився з базовими принципами командної розробки.
100. Отримав загальне уявлення про повний цикл розробки програмного забезпечення.