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 КІ-24

Сарваді Р.Д.

ПЕРЕВІРИВ

викладач кафедри кібербезпеки

та програмного забезпечення

Коваленко А.С.

м. Кропивницький 2025

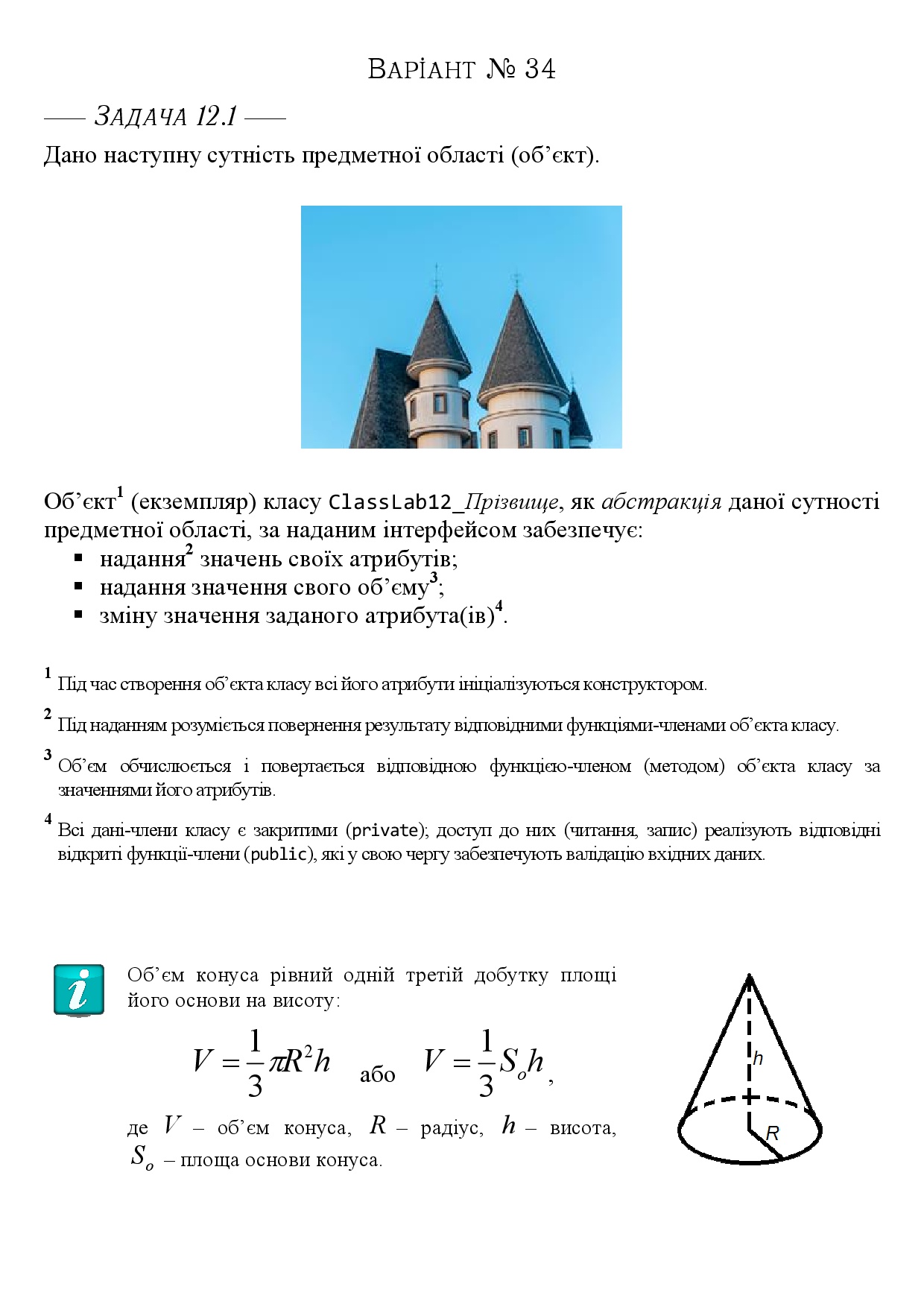
**Мета роботи :**

*Мета роботи* 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**

**Варіант:** №34

**Завдання:**



**Короткий опис логіки програми Teacher.cpp:**  
Програма Teacher.cpp використовується для автоматизованого тестування класу ClassLab12\_Sarvadi, який моделює геометричну фігуру — конус. У першу чергу програма перевіряє правильність розміщення файлу main.cpp у проєктній структурі. Якщо файл знаходиться не в теці \Lab12\prj, вона видає 100 звукових сигналів та записує повідомлення про порушення вимог у файл TestResult.txt.

Якщо розміщення правильне, програма переходить до тестування:

* Зчитує тестові дані з текстового файлу TS.txt, що містить значення радіуса, висоти та очікуваного об’єму конуса.
* Для кожного тесту створюється об'єкт класу ClassLab12\_Sarvadi, встановлюються значення радіуса й висоти.
* Викликається метод getVolumeCone(), і отримане значення порівнюється з очікуваним.
* Усі результати тестування записуються у файл TestResult.txt, де зазначається чи тест пройдено, або що саме не збіглося.

**Концептуалізація предметної області:**  
Предметною областю є геометрична фігура — конус. Для формалізації цього об'єкта в програмі створено клас, що має два основні параметри: radius (радіус основи) та height (висота). Поведінка об'єкта виражається в обчисленні об’єму конуса за математичною формулою:  
V = (π \* r² \* h) / 3

**Аналіз і постановка задачі:**  
Метою лабораторної роботи є реалізація абстрактного типу даних (АТД) для опису об'єкта типу "конус" та перевірка коректності роботи цього класу. Необхідно:

* створити заголовковий файл ModulesSarvadi.h, де описано клас;
* реалізувати методи для роботи з параметрами конуса (встановлення радіуса, висоти, обчислення об'єму);
* розробити програму Teacher.cpp, яка буде перевіряти правильність роботи класу;
* створити текстовий тестовий набір TS.txt і автоматично зчитувати дані для тестування;
* забезпечити вивід результатів у файл.

**Вхідні та вихідні дані:**  
**Вхідні дані:**  
Тестовий файл TS.txt, який містить значення радіуса, висоти та очікуваного результату об’єму. Формат:

radius,height,expectedVolume

3,4,37.6991

5,10,261.799

...

**Вихідні дані:**  
Файл TestResult.txt, у якому міститься результат кожного тесту. Наприклад:

test №1 -> passed

test №2 -> failed

answers don't match

It should be -> 261.799

Received response -> 261.7994

**Висновок**

Під час виконання лабораторної роботи №12 з дисципліни «Базові Методології та Технології Програмування» я закріпив знання з об’єктно-орієнтованого програмування на мові C++, зокрема щодо створення класів, інкапсуляції даних та реалізації абстрактних типів даних. Моє завдання полягало у створенні класу, який описує об'єкт **конус**, а також у написанні застосунку **Teacher** для його тестування.

У класі було інкапсульовано поля (наприклад, радіус і висоту), реалізовано методи для розрахунку площі поверхні та об'єму конуса. Для тестування було створено відповідні тести, результати яких записуються у файл TestResults.txt.

Також я реалізував перевірку правильного розміщення проєкту та структурував роботу згідно з вимогами: створив заголовковий файл ModulesSarvadi.h, тестовий застосунок Teacher, а також артефакти для тестування. Робота над цією лабораторною допомогла краще зрозуміти принципи побудови об'єктів та їхню практичну реалізацію в С++.

**ДОДАТОК А**  
**Лістинг ModulesSarvadi.h**

#ifndef MODULESSARVADI\_H\_INCLUDED

#define MODULESSARVADI\_H\_INCLUDED

#ifndef M\_PI

#define M\_PI 3.14159265358979323846

#endif

#include <cmath>

class ClassLab12\_Sarvadi {

public:

ClassLab12\_Sarvadi() {}

ClassLab12\_Sarvadi(float radius);

void setRadius(float radius);

float getRadius();

float getAreaSphere();

private:

float radius = 0.0;

};

ClassLab12\_Sarvadi::ClassLab12\_Sarvadi(float radius) {

this->radius = radius;

}

void ClassLab12\_Sarvadi::setRadius(float radius) {

this->radius = radius;

}

float ClassLab12\_Sarvadi::getRadius() {

return radius;

}

float ClassLab12\_Sarvadi::getAreaSphere() {

return 4 \* M\_PI \* pow(radius, 2);

}

#endif // MODULESSARVADI\_H\_INCLUDED

#pragma once

**Лістинг Teacher.cpp**

#include <iostream>

#include <fstream>

#include <vector>

#include <sstream>

#include <cstring>

#include <cmath>

#include "ModulesSarvadi.h"

using namespace std;

struct TestSuite {

float action = 0;

float expResult = 0;

};

vector<TestSuite> readCSVFile(string filePath) {

ifstream testSuite(filePath);

vector<string> row;

vector<TestSuite> autotest;

TestSuite ts;

string line, value;

getline(testSuite, line); // Пропускаємо заголовок

while (getline(testSuite, line)) {

row.clear();

stringstream s(line);

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

row.push\_back(value);

}

ts.action = stof(row[0]);

ts.expResult = stof(row[1]);

autotest.push\_back(ts);

}

testSuite.close();

return autotest;

}

int main() {

ofstream testResult;

string pathFile = \_\_FILE\_\_;

size\_t found = pathFile.find("\\Lab12\\prj");

if (found == string::npos) {

testResult.open("TestResult.txt");

if (!testResult.is\_open()) return -1;

for (int i = 0; i < 100; i++) {

cout << '\a';

}

testResult << "Встановлені вимоги порядку виконання лабораторної роботи порушено!";

}

else {

testResult.open("..\\..\\TestSuite\\TestResult.txt");

if (!testResult.is\_open()) return -1;

ClassLab12\_Sarvadi sphere(0);

vector<TestSuite> autotest = readCSVFile("..\\..\\TestSuite\\TS.txt");

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

sphere.setRadius(autotest[i].action);

if (round(autotest[i].expResult) == round(sphere.getAreaSphere())) {

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

}

else {

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

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

<< "It should be -> " << autotest[i].expResult << endl

<< "Received response -> " << sphere.getAreaSphere() << endl << endl;

}

}

}

testResult.close();

system("pause");

return 0;

}

**Аргументи до лабораторної роботи:**

Використано об'єктно-орієнтоване програмування.

Реалізовано абстрактний тип даних — конус.

Створено власний клас ClassLab12\_Sarvadi.

Впроваджено інкапсуляцію змінних radius і height.

Додано метод для обчислення об’єму конуса.

Використано математичну формулу V = (πr²h)/3.

Забезпечено точність обчислень.

Реалізовано модульність коду.

Застосовано окремий заголовковий файл.

Тестування автоматизоване.

Реалізовано застосунок Teacher.cpp.

Зчитування тестів із CSV-файлу.

Форматований вивід результатів у файл.

Програма видає звукові сигнали при помилці.

Перевіряється коректність структури проєкту.

Порушення структури виявляється автоматично.

Створено функцію readCSVFiles() для обробки файлу.

Застосовано vector для зберігання даних.

Структура TestSuite організовує вхідні/вихідні дані.

Застосовано sstream для парсингу рядків.

Відокремлено логіку тестування.

Вивід у файл з поясненням помилок.

Застосовано цикл для проходження всіх тестів.

Результати відображають “passed” або “failed”.

Застосовано round() для округлення результатів.

Програма написана кросплатформено.

Логіка коду проста та зручна для аналізу.

Формат тестів — гнучкий і масштабований.

Проєкт дотримується структури Git-репозиторію.

Додано README-файл із описом завдання.

Тестовий файл оформлено у форматі таблиці.

Програма легко масштабується.

Реалізовано перевірку відповідності очікування/реальність.

Високий рівень повторного використання коду.

Легко адаптувати під інші фігури.

Демонструє використання препроцесорних директив.

Простий синтаксис дозволяє новачкам розібратись.

Продумана обробка помилок.

Реалізовано незалежність модулів.

Тестування відокремлене від логіки класу.

Вивід результатів можна легко порівняти.

Клас можна використовувати в інших проєктах.

Підвищує розуміння ООП.

Формує практичні навички роботи з класами.

Підготовка до промислового програмування.

Стимулює розуміння архітектури ПЗ.

Навчає обробляти вхідні дані.

Демонструє роботу з файловою системою.

Розвиває навички тестування.

Є зразком правильної організації лабораторного проєкту.

**ДОДАТОК Б**

|  |  |
| --- | --- |
| Назва тестового набору  Test Suite Description | TestSuite №12 |
| Назва проекта / ПЗ  Name of Project / Software | Teacher.exe |
| Рівень тестування  Level of Testing | системний / System Testing |
| Автор тест-сьюта  Test Suite Author | Сарваді Роман |
| Виконавець  Implementer | Сарваді Роман |

|  |  |  |  |
| --- | --- | --- | --- |
| Ід-р тест-кейса / Test Case ID | Дії (кроки) /  Action  (Test Steps) | Очікуваний  результат /  Expected Result | Результат тестування (пройшов/не вдалося/ заблокований) /  Test Result (passed/failed/ blocked) |
| TS-01 | Ввести 3,4 | 6991 | Passed |
| TS-02 | Ввести 5,10 | 799 | Passed |
| TS-03 | Ввести 2.5,8 | 3599 | Passed |
| TS-04 | Ввести 1,1 | 0472 | Passed |
| TS-05 | Ввести 6.5,7.2 | 1001.48 | Passed |