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

ВИКОНАВ

студент академічної групи

КІ-21-2

Берестенко Д.О.

ПЕРЕВІРИВ

ст. викладач кафедри кібербезпеки   
та програмного забезпечення

Усік П.С.

Кропивницький – 2022

**Лабораторна робота №12**

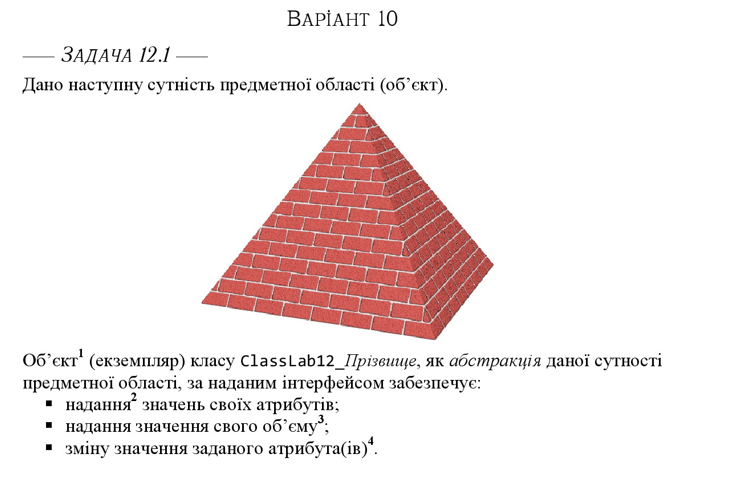
**Тема:** програмна реалізація абстрактних типів даних.

**Мета роботи:** полягає у набутті грунтовних вмінь і практичних навичок об'єктного аналізу й проектування, створення класів C++ та тестування їх екземплярів, використання препроцесорних директив, макросів і макрооператорів під час реалізації програмних засобів у кросплатформовому середовищі Code::Blocks.

**Завдання:**

1. Як складову заголовкового файлу ModulesПрізвище.h розробити клас ClassLab12\_Прізвище – формальне представлення абстракції сутності предметної області (об’єкта) за варіантом, – поведінка об'єкта реалізовує розв'язування задачі 7.1.
2. Реалізувати додаток Teacher, який видає 100 звукових сигналів і в текстовий файл TestResults.txt записує рядок “Встановлені вимоги порядку виконання лабораторної роботи порушено!”, якщо файл проекта main.срр під час його компіляції знаходився не в \Lab12\prj, інакше – створює об’єкт класу ClassLab12\_Прізвище із заголовкового файлу ModulesПрізвище.h та виконує його unit-тестування за тест-сьютом(ами) із \lab12\TestSuite, протоколюючи результати тестування в текстовий файл \lab12\TestSuite\TestResults.txt.

**Варіант** **10**



**Аналіз вимог до програмного модуля:**

Модуль має складатися з опису класу Class Lab12\_Berestenko, який є формальним представленням абстракції піраміди. Також екземпляри класу мають забезпечувати за наданим інтерфейсом надання своїх атрибутів (повернення значень певних атрибутів за допомогою відповідних функцій), надання значення свого об’єму (яке обчислюжться та повертається відповідним методом за значеннями атрибутів класу) та зміну значення заданих атрибутів (вхідні дані валідуються). Реалізувати інкапсулювання даних-членів класу, доступ до них відбувається за допомогою відповідних методів.

1. **Формат тест-кейса**

Тест-кейси для функцій зміни значення атрибутів складаються із вхідного значення та результату, розділених будь-якою кількістю відступів (пробіли, табуляції тощо). Кожний тест-кейс для зручності знаходиться в окремому рядку. Програма зчитує кожне значення як слово, що вирішує проблему з довільною кількістю відступів між значеннями. Вхідне значення та результат зчитуються по-черзі.

Тест-кейси для функції надання значення об’єму складаються з двох вхідних значеннь та результату, розділених будь-якою кількістю відступів .В іншому вони схожі з тест-кейсами попередніх функцій. Спочатку почерзі зчитуються перші три значення, які відповідають вхідним даним, після чого зчитується результат.

**Аналіз вимог до програмного засобу:**

Програмний засіб Teacher має перевіряти знаходження вихідного файлу під час компіляції, і якщо він не знаходиться в (під-)директорії lab12/pry/ видавати 100 звукових сигналів, створювати у директоріїї вихідного файлу текстовий файл зі змістом про те, що умови виконання лабораторної роботи порушено, інакше – створювати об’єкт спроектованого класу та виконувати його unit-тестування за тест-сьютами з lab12\TestSuite\, протоколюючи результати у файл TestResults.txt в цій директорії.

**Проектування програмного засобу:**

Програмний засіб за допомогою макросу \_\_FILE\_\_ отримує під час компіляції повний шлях до вихідного файлу main.cpp, для якого перевіряється, чи знаходиться у потрібній директорії. Якщо ввимога не виконується, в стандартний потік виведення записується 100 символів “\a”, після чого у головній директорії створюється файл TestResults.txt, у який записується повідомлення про порушення вимог. В інакшому випадку створюється елемент класу ClassLab12\_Berestenko, після чого відбувається unit-тестування функцій зміни атрибутів класу та функції надання значення об’єму. Для першого випадку функції передається перше значення тест-кейсу відповідного тест-сьюту, після чого результат функції надання значення атрибуту порівнюється зі значенням очікуваного результату. Для методу об’єму вхідні дані задаються за допомогою функцій зміни атрибутів, а результат роботи методу порівнюється з відповідним очікуваним результатом. Результати порівнянь протоколюються у файл TestResults.txt.

Лістинг вихідного коду заголовкового коду Modules\_Berestenko.h наведений у Додатку А.

Лістинг вихідного коду проекту Teacher наведено у Додатку Б.

Вміст файлу TestResults.txt наведений у Додатку В.

Результати системного тестування додатка наведені у Додатку Г.

**Тестові артефакти:**

Вхідні значення та очікувані записані в текстовому файлі й приведені до спільної структури, що надалі буде використовуватись додатком Teacher.exe. Протоколювання читання здійснюється 3-ма умовами: у випадку, якщо зчитування неможливе, до текстового файлу TestResults.txt записується “Встановлені вимоги порядку виконання лабораторної роботи порушено!” та завершується робота застосунку; якщо зчитування вдале - відбувається Unit-тестування. Вміст файлу TS.txt:

piramidHeight: 15

piramidLenght: 6

piramidVolume: 77.942286

piramidHeight: 5

piramidLenght: 20

piramidVolume: 288.675135

piramidHeight: 10

piramidLenght: 8

piramidVolume: 92.376043

piramidHeight: 10

piramidLenght: 4

piramidVolume: 23.094011

piramidHeight: 14

piramidLenght: 7

piramidVolume: 99.015571

piramidHeight: 11

piramidLenght: 2

piramidVolume: 6.350853

piramidHeight: 12

piramidLenght: 5

piramidVolume: 43.301270

piramidHeight: 12

piramidLenght: 10

piramidVolume: 173.205081

piramidHeight: 18

piramidLenght: 11

piramidVolume: 314.367222

piramidHeight: 21

piramidLenght: 14

piramidVolume: 594.093427

piramidHeight: 22

piramidLenght: 16

piramidVolume: 812.909179

piramidHeight: 10

piramidLenght: 15

piramidVolume: 324.759526

piramidHeight: 17

piramidLenght: 21

piramidVolume: 1082.098742

**Висновок:** дана лабораторна роботта була націлена на набуття навичок об'єктного аналізу й проектування, створення класів C++ та тестування їх екземплярів, використання препроцесорних директив, макросів і макрооператорів мовою програмування C++.

Для об’єктного аналізу було запропоновано акваріум, у якого в залежності від вимог потрібно було виділити значущі властивості та спроектувати його поведінку. Формально абстракцію предметної області було реалізовано за допомогою класу мовою програмування C++. У створеному класі ClassLab12\_Berestenko атрибути були інкапсульовані (описані у секції private), а методи класу, що відповідають за поведінку екземпляру класу та роботу з атрибутами – на рівні public.

Замість двох констукторів (без параметрів та з трьома обов'язковими) було створено один конструктор, у якому реалізованні значення за замовчуванням. Клас також повинен мати метод надання об’єму акваріуму, який вираховується за формулою. Таким чином, значущими атрибутами об’єкту є його довжина, ширина та висота.

За вимогами кожному атрибуту відповідає дві функції: одна надає значення атрибуту (повертає його значення), інша – змінює значення відповідного атрибуту, валідуючи це значення (усі атрибути є мірами довжини, тож не можуть бути меншими за 0). Метод надання об’єму повертає рельзультат множення усіх атрибутів.

Після програмної реалізації класу у заголовковому файлі Modules\_Berestenko.h, що був створений під час лабораторної роботи 8, було створено консольний додаток Teacher, який призначений для unit-тестування класу та має кілька особливостей.

По-перше, якщо додаток скомпільовано не в (під-)директорії lab12\prj\, він має видавати 100 звукових сигналів, створювати у своїй директорії текстовий файл TestResults.txt та записувати в нього повідомлення про те, що вимоги до проектування ПЗ порушено. Зробити це можна за допомогою макросу \_\_FILE\_\_, який містить рядок з повним шляхом до вихідного файлу, який був скомпільований. Щоб перевірити, чи знаходиться він у потрібній директорії, цей підрядок шукається у рядку шляху. Якщо підрядок не знайдено – відбуваються вищезазначені дії та процес повертає код -1, інакше відбувається unit-тестування.

ДОбраною структурою TestSuite став текстовий файл TS.txt, в кожному рядку якого, стала кількість перших символів відведена для дескрипції значення, записаного далі. Таким чином, при зчитуванні даних з файлу, з рядка відкидаються перші 12 символів. Зчитування з TS.txt відбувається в рядок bufLine методом getline(), починаючи з 13 символа в рядку. Відбувається окремо для радіуса та площі кола.

Для зручної навігації та аналізу структури коду, додаткові локальні функції описано в файлі AdditionFunctions.h та перед функцією main у застосунку Teacher.

Результати тестування було записано у текстовий файл TestResults.txt у вигляді таблиці для зручного сприйняття інформації.Наступним кроком є створення екземпляру класу ClassLab12\_Berestenko, після чого відбувається тестування. Хоч клас і містить 8 методів, має сенс перевіряти лише 4 з них – функції зміни атрибутів класу та надання об’єму. Функції зміни атрибутів мають однакову структуру, тож було створено окрему функцію, яка приймає в якості аргументів посилання на тестований метод, посилання на відповідний метод надання значень атрибутів (прямий доступ до значень атрибутів заблокований), тестований об’єкт, шлях до тест-сьюту та опціональний заголовок результату тестування окремого методу. З одного боку це зручно, бо не потрібно повторювати код, який тестує одну функцію. В якомусь сенсі ця функція використовує функціяю для тестування в якості змінної.

Функція спочатку створює у теці TestSuite файл TestResults.txt, потім відкриває тест-сьют. Особливістю файлового потоку є можливість зчитування слів (послідовностей символів, не містящих відступів) незалежно від кількості символів відступу (пробіл, табуляція тощо). Далі зчитане слово перетворюється на число з плаваючою комою (якщо це неможлиіо – на число 0), передається у функцію зміни атрибуту та потім результат функції надання атрибуту порівнюєьтся з наступним словом, перетвореним на чилсо з плаваєчою точкою. Для коректної конвертації рядку у число з плаваючою комою використовується кодування CP1251.

Результати тестування належно протоколюются у файл з результатами тестування за допомогою маніпулятор вводу-виведення. Кількість тест-кейсів необмежена програмно, тому зчитування відбувається до моменту, коли настане кінець файлу. Якщо будь-який файл немможливо відкрити, повідомлення про це виводиться у консоль, та тестування методу не відбувається, а додаток продовжує роботу. Також через ідентичність вимог до функцій зміни атрибутів для них використовується один тест-сьют.

Тестування методу надання значення об’єму відбувається аналогічно, однак воно виконується іншою функцією, яка приймає тільки одну функцію та має три значення вхідних даних та одну очікуваного результату для кожного тест-кейсу. Кожне вхідне значення надається об’єкту за допомогою функцій зміни атрибутів.

**ДОДАТОК А**

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

| TEST CASE 1 |

| (input) Piramid height: 15 м |

| (input) Piramid side length: 6 м |

| (output) Piramid volume: 77.942286 м^2 |

| (expected) Piramid volume: 77.942286 м^2 |

| TEST CASE RESULT: true |

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

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

| TEST CASE 2 |

| (input) Piramid height: 5 м |

| (input) Piramid side length: 20 м |

| (output) Piramid volume: 288.675135 м^2 |

| (expected) Piramid volume: 288.675135 м^2 |

| TEST CASE RESULT: true |

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

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

| TEST CASE 3 |

| (input) Piramid height: 10 м |

| (input) Piramid side length: 8 м |

| (output) Piramid volume: 92.376043 м^2 |

| (expected) Piramid volume: 92.376043 м^2 |

| TEST CASE RESULT: true |

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

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

| TEST CASE 4 |

| (input) Piramid height: 10 м |

| (input) Piramid side length: 4 м |

| (output) Piramid volume: 23.094011 м^2 |

| (expected) Piramid volume: 23.094011 м^2 |

| TEST CASE RESULT: true |

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

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

| TEST CASE 5 |

| (input) Piramid height: 14 м |

| (input) Piramid side length: 7 м |

| (output) Piramid volume: 99.015571 м^2 |

| (expected) Piramid volume: 99.015571 м^2 |

| TEST CASE RESULT: true |

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

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

| TEST CASE 6 |

| (input) Piramid height: 11 м |

| (input) Piramid side length: 2 м |

| (output) Piramid volume: 6.350853 м^2 |

| (expected) Piramid volume: 6.350853 м^2 |

| TEST CASE RESULT: true |

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

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

| TEST CASE 7 |

| (input) Piramid height: 12 м |

| (input) Piramid side length: 5 м |

| (output) Piramid volume: 43.301270 м^2 |

| (expected) Piramid volume: 43.301270 м^2 |

| TEST CASE RESULT: true |

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

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

| TEST CASE 8 |

| (input) Piramid height: 12 м |

| (input) Piramid side length: 10 м |

| (output) Piramid volume: 173.205081 м^2 |

| (expected) Piramid volume: 173.205081 м^2 |

| TEST CASE RESULT: true |

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

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

| TEST CASE 9 |

| (input) Piramid height: 18 м |

| (input) Piramid side length: 11 м |

| (output) Piramid volume: 314.367222 м^2 |

| (expected) Piramid volume: 314.367222 м^2 |

| TEST CASE RESULT: true |

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

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

| TEST CASE 10 |

| (input) Piramid height: 21 м |

| (input) Piramid side length: 14 м |

| (output) Piramid volume: 594.093427 м^2 |

| (expected) Piramid volume: 594.093427 м^2 |

| TEST CASE RESULT: true |

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

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

| TEST CASE 11 |

| (input) Piramid height: 22 м |

| (input) Piramid side length: 16 м |

| (output) Piramid volume: 812.909179 м^2 |

| (expected) Piramid volume: 812.909179 м^2 |

| TEST CASE RESULT: true |

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

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

| TEST CASE 12 |

| (input) Piramid height: 10 м |

| (input) Piramid side length: 15 м |

| (output) Piramid volume: 324.759526 м^2 |

| (expected) Piramid volume: 324.759526 м^2 |

| TEST CASE RESULT: true |

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

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

| TEST CASE 13 |

| (input) Piramid height: 17 м |

| (input) Piramid side length: 21 м |

| (output) Piramid volume: 1082.098742 м^2 |

| (expected) Piramid volume: 1082.098742 м^2 |

| TEST CASE RESULT: true |

**ДОДАТОК Б**

#ifndef MODULES\_BERESTENKO\_H\_INCLUDED

#define MODULES\_BERESTENKO\_H\_INCLUDED

#include <cmath>

class ClassLab12\_Berestenko{

public:

ClassLab12\_Berestenko(){}

ClassLab12\_Berestenko(double height, double side\_length);

void setPiramidHeight(double height);

double getPiramidHeight();

void setPiramidLength(double side\_length);

double getPiramidLength();

private:

double height = 1.0;

double side\_length = 4.0;

};

ClassLab12\_Berestenko::ClassLab12\_Berestenko(double height, double side\_length){

this->height = height;

this->side\_length = side\_length;

}

void ClassLab12\_Berestenko::setPiramidHeight(double height){

this->height = height;

}

void ClassLab12\_Berestenko::setPiramidLength(double side\_length){

this->side\_length = side\_length;

}

double ClassLab12\_Berestenko::getPiramidHeight(){

return height;

}

double ClassLab12\_Berestenko::getPiramidLength(){

return side\_length;

}

double getPiramidVolume(double height,double side\_length){

return ((((side\_length \* side\_length \* sqrt(3))/4) \* height)/3);

}

#endif // MODULES\_BERESTENKO\_H\_INCLUDED

**ДОДАТОК В**

#ifndef ADITIONFUNC\_H\_INCLUDED

#define ADITIONFUNC\_H\_INCLUDED

#include "Modules\_Berestenko.h"

#include <iostream>

#include <fstream>

#include <iomanip>

using namespace std;

bool fileInDirectory()

{

string cpp = \_\_FILE\_\_;

size\_t found = cpp.find("\\lab12\\prj");

if (found == string::npos) {

ofstream resFile("../TestSuite/TestResults.txt");

resFile << "Встановлені вимоги порядку виконання лабораторної роботи порушено!" << endl;

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

cout << "\a";

}

resFile.close();

return false;

}

return true;

}

bool filesIsOpen(ofstream &wfile, ifstream &rfile)

{

if (!rfile.is\_open() || !wfile.is\_open() || !rfile || !wfile) {

ofstream resFile("../TestSuite/TestResults.txt");

resFile << "Встановлені вимоги порядку виконання лабораторної роботи порушено!" << endl;

resFile.close();

return false;

}

return true;

}

void declareTestSuiteResults(ofstream &resFile, ClassLab12\_Berestenko pir, string outArea, string expArea, int index)

{

resFile << "----------------------------------------------------" << endl;

resFile << "| TEST CASE " << setw(2) << index << setw(38) << "|" << endl;

resFile << "| (input) Piramid height: " << setw(17) << pir.getPiramidHeight() << " м |" << endl;

resFile << "| (input) Piramid side length: " << setw(12) << pir.getPiramidLength() << " м |" << endl;

resFile << "| (output) Piramid volume: " << setw(17) << outArea << " м^2 |" << endl;

resFile << "| (expected) Piramid volume: " << setw(17) << expArea << " м^2 |" << endl;

resFile << "| TEST CASE RESULT: " << setw(20) << boolalpha << (expArea.compare(outArea) == 0) << " |" << endl;

resFile << "----------------------------------------------------" << endl;

}

#endif // ADITIONFUNC\_H\_INCLUDED

**ДОДАТОК В**

#include "AditionFunc.h"

using namespace std;

int main()

{

string tsDirPath = "./";

string testResPath = "TestResults.txt";

system("chcp 1251 & cls");

string path = \_\_FILE\_\_;

if (path.find("lab12\\prj\\") == -1) {

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

cout << "\a";

}

ofstream testRes(testResPath);

if (testRes.is\_open()) {

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

testRes.close();

}

else {

cout << "Помилка при роботі з файлом " << testResPath << "." << endl;

}

return -1;

}

ifstream mainFile("main.cpp");

if (mainFile.is\_open()) {

tsDirPath = "../../Testsuite/";

mainFile.close();

}

else {

tsDirPath = "../Testsuite/";

}

testResPath = tsDirPath+testResPath;

ofstream testRes(testResPath, ios::out);

if (testRes.is\_open()) testRes.close();

string TSPath = tsDirPath+"TS.txt";

ifstream caseFile(TSPath);

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

cout << TSPath << " is not opened!";

return -1;

}

ofstream resultFile(testResPath);

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

cout << testResPath << " is not opened!";

return -1;

}

ClassLab12\_Berestenko piramid;

for (int i = 1; i < 14; i++) {

string piramidHeight;

string piramidLength;

string piramidVolume;

string bufLine;

getline(caseFile, bufLine);

piramidHeight = bufLine.erase(0, 15);

getline(caseFile, bufLine);

piramidLength = bufLine.erase(0, 15);

getline(caseFile, bufLine);

piramidVolume = bufLine.erase(0, 15);

piramid.setPiramidHeight(stof(piramidHeight));

piramid.setPiramidLength(stof(piramidLength));

bufLine = to\_string(getPiramidVolume(piramid.getPiramidHeight(),piramid.getPiramidLength()));

declareTestSuiteResults(resultFile,piramid,bufLine,piramidVolume,i);

}

caseFile.close();

resultFile.close();

system("pause");

return 0;

}