Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

# ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 12

з навчальної дисципліни “Базові методології та технології програмування”

ПРОГРАМНА РЕАЛІЗАЦІЯ АБСТРАКТНИХ ТИПІВ ДАНИХ

ВИКОНАВ

студент академічної групи

КІ-21-1 Назаревський М. В.

ПЕРЕВІРИВ

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 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.

## Варіант 2

**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 якого 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\, проколюючи результати тестування в текстовий файл \Lab12\TestSuite\TestResults.txt



Рисунок 1 - Умова задачі 12.1

**ХІД РОБОТИ**

### **Аналіз задачі**

За умовою завдання, дано візуалізацію об’єкта, структуру якого можна інтерпретувати як “Аквариум”. Однак, спираючись на вимоги щодо значень множини вхідних та вихідних даних, достатньо обмежитись концепцією більш абстрактного об’єкта - геометричної фігури прямокутний паралелепіпед, без верхньої основи.

Тому, визначення всіх властивостей описується значенням висоти, ширини та довжини. Обмеження та допущення щодо вхідних даних зводяться до пошуку найбільшої та найменшої в світі структури, схожою з обраною концепцією, адже маніпулювати довжинами у нескінченних та від’ємних межах - неможливо.

### **Вимоги до програмного модуля:**

Властивості об’єкта (ширина, висота, довжина - в якості полів, площа - в якості метода) оголошуються на рівні private. Методи отримання значень з рівня private (функції мають префікс get в ідентифікаторі) та методи ініціалізації полів класу (функції, що мають префікс set в ідентифікаторі) оголошуються на рівні public.

Поля класу за замовчуванням ініціалізуються *1* (в сантиметрах), що надалі використовуються як обмеження або виявлення некоректної ініціалізації.

Процедури користувацького визначення радіуса та висоти мають коректно зчитувати як десяткову крапку так і кому, а також відкидати присвоєння нового значення, якщо аргумент значення не приведено до форми стандартного типу float з точністю до 6 цифр після коми.

### **Артефакти детального проектування програмного модуля:**

Приватні поля класу:

pipeHeight - висота прямокутного паралелепіпеда

pipeLength - довжина прямокутного паралелепіпеда

pipeWidth - ширина прямокутного паралелепіпеда

Приватні методи класу:

pipeParallelepiped() - функція, в якості аргументів отримує висоту, довжину та ширину прямокутного паралелепіпеда та повертає значення його площі;

Публічні методи класу:

getPipeHeight(): повертає значення приватної змінної pipeHeight;

getPipeLenght(): повертає значення приватної змінної pipeLenght;

getPipeWidth(): повертає значення приватної змінної pipeWidth;

getPipeParallelepiped(): видає значення приватного методу pipeParallelepiped();

setPipeHeight(): функція приймає два аргументи: рядок, та адресу змінної типу float. В рядку всі коми замінюються на крапки. За допомогою бібліотеки <sstream>, виконується перевірка отриманого рядка на відповідність числовому значенню типу float. У випадку істинності виразу, приватне поле pipeHeight ініціалізується даним рядком, приведеним до типу float методом stof(), інакше - ініціалізується значенням за замовчуванням.

setPipeLenght(): ідентично до setPipeHeight(), однак ініціалізується приватне поле pipeLenght;

setPipeWidth(): ідентично до setPipeHeight(), однак ініціалізується приватне поле pipeWidth;

### **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 2-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:

pipeHeight: 20

pipeLength: 15

pipeWidth: 10

pipeParallelepiped: 3000.000000

pipeHeight: 60.0

pipeLength: 150.0

pipeWidth: 60.0

pipeParallelepiped: 540000.000000

pipeHeight: 3,3

pipeLength: 9,5

pipeWidth: 13,1

pipeParallelepiped: 410.684998

pipeHeight: 1234

pipeLength: 73126

pipeWidth: 8196478

pipeParallelepiped: 1.000000

pipeHeight: idk

pipeLength: word

pipeWidth: lol

pipeParallelepiped: 1.000000

pipeHeight: -2

pipeLength: -12

pipeWidth: -4

pipeParallelepiped: 1.000000

pipeHeight: 2

pipeLength: 40

pipeWidth: 50

pipeParallelepiped: 4000.000000

pipeHeight: 43,21

pipeLength: 81,78

pipeWidth: 12,523

pipeParallelepiped: 44252.695312

pipeHeight: 60

pipeLength: 35

pipeWidth: 12

pipeParallelepiped: 25200.000000

pipeHeight: 29

pipeLength: 143.5

pipeWidth: 48.32

pipeParallelepiped: 201083.671875

### **Аналіз та архітектура завдання 12.2:**

На початку роботи застосунку, виконується перевірка, зазначена у вимогах завдання 12.2. Робота застосунку припиняється, якщо файл main.cpp не знайдено в теці ../prj.

Створюється об’єкт класу ClassLab12\_Nazarevskyi, файл TS.txt відкривається для зчитування, файл TestResults.txt відкривається для дозапису (попередній вміст очищається, при відкритті файлу). Відбувається перевірка на наявність даних текстових файлів в директорії та чи відкриті вони на момент роботи даної функції. Якщо хоча б одна умова хибна - робота застосунку припиняться.

Цикл на 10 ітерацій, призначений для декларування процесу тестування, виконує наступний порядок дій:

* оголошення рядків, призначених для збереження висоти, довжини, ширини та площі прямокутного паралелепіпеда . Буферний рядок, що використовуються в проміжних операціях. Змінна типу float, що не має бути ініціалізовано.
* зчитування з TS.txt в рядок bufLine методом getline(), починаючи з 13 символа в рядку. Відбувається окремо для висоти, довжини, ширини та площі прямокутного паралелепіпеда.
* висота, довжина та ширина ініціалізуються публічними методами, а площа що отримана публічним методом та приведена до типу string записується в змінну bufLine.
* виконується форматоване виведення значень та отриманого результату Тест Кейсів у файл TestResults.txt.

# 

# ВИСНОВКИ

Мета полягала 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.

Під час процесу виконання завдань, виникли наступні спостереження щодо потреб лабораторної роботи: не дивлячись на кінцевий результат реалізації кожного методу та процедури, що не виділяються достатньою складністю, метою лабораторної роботи стала демонстрація навичок коректної абстракції сутності предметної. Розглядаючи точніше - абстрагування мало бути відповідним до вимог задачі, тобто мати обмеження щодо рівня абстрагування (деталізація об’єкта була мінімальною).

Згідно з виданого завдання, абстракція сутності аквариуму, було зведено до простої геометричної фігури - прямокутний паралелепіпед без верхньої основи. Змінних довжини, висоти, ширини та метода отримання площі було достатньо для представлення властивостей прямокутного паралелепіпеду.

Для кожного поля (приватної змінної класу) відведено два метода, визначені порядком лабораторної: функція отримання значення з приватного поля (метод повертає вміст змінної) та функція ініціалізації даних полів (містить перевірку на відповідність типу та порівняння з визначеними обмеженнями. Присвоює приватному полю коректне значення).

Я оголосив площу як закритий метод з поверненням результату розрахунків. Дана процедура є прив’язаною функцією отримання площі, що також є способом викликати закриту процедуру розрахунку площі.

Концепція роботи метода ініціалізації ширини, довжини та висоти прямокутного паралелепіпеда: в якості аргументів, функція отримує ініціалізований рядок та адресу оголошеного числа типу float. В тілі оголошується змінна типу char, рядку присвоюються цей же рядок, в якому коми заміщені крапками і відбувається визначення змінної типу stringstream, залежної від рядка. Якщо виконується умова побітової перевірки вхідного рядка з типом змінної float і приведена до цього типу змінна string не порушує обмежень - приватне поле ініціалізується приведеним до типу float рядка.

Обраною структурою TestSuite став текстовий файл, 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 n символів.

Для зручної навігації та аналізу структури коду, додаткові локальні функції описано в файлі TestFunctions.h.

В цьому файлі присутні три функції:

* fileInDirectory(): відповідає за перевірку наявності файлу main.cpp в теці ../prj під час компіляції проекту. Містить реалізацію вимог завдання №2.
* filesIsOpen(): виконує повну перевірку можливості до редагування та наявність відкритих текстових файлі TS.txt та TestResult.txt. Якщо хоча б одна умова хибна - в TestResult.txt записується повідомлення про порушення вимог лабораторної роботи.
* declareTestSuiteResults(): отримує всі необхідні аргументи, в якості значення, що використовуються для декларування результату модульного тестування за описаним тест кейсом.

**ДОДАТОК А**

**…/TestSuite/TestResults.txt**

TEST CASE 1

(input) Pipe height: 20 cм

(input) Pipe length: 15 cм

(input) Pipe width: 10 cм

(output) Pipe parallelepiped: 3000.000000 cм^3

(expected) Pipe parallelepiped: 3000.000000 cм^3

TEST CASE RESULT: true

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

TEST CASE 2

(input) Pipe height: 60 cм

(input) Pipe length: 150 cм

(input) Pipe width: 60 cм

(output) Pipe parallelepiped: 540000.000000 cм^3

(expected) Pipe parallelepiped: 540000.000000 cм^3

TEST CASE RESULT: true

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

TEST CASE 3

(input) Pipe height: 3.3 cм

(input) Pipe length: 9.5 cм

(input) Pipe width: 13.1 cм

(output) Pipe parallelepiped: 410.684998 cм^3

(expected) Pipe parallelepiped: 410.684998 cм^3

TEST CASE RESULT: true

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

TEST CASE 4

(input) Pipe height: 1 cм

(input) Pipe length: 1 cм

(input) Pipe width: 1 cм

(output) Pipe parallelepiped: 1.000000 cм^3

(expected) Pipe parallelepiped: 1.000000 cм^3

TEST CASE RESULT: true

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

TEST CASE 5

(input) Pipe height: 1 cм

(input) Pipe length: 1 cм

(input) Pipe width: 1 cм

(output) Pipe parallelepiped: 1.000000 cм^3

(expected) Pipe parallelepiped: 1.000000 cм^3

TEST CASE RESULT: true

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

TEST CASE 6

(input) Pipe height: 1 cм

(input) Pipe length: 1 cм

(input) Pipe width: 1 cм

(output) Pipe parallelepiped: 1.000000 cм^3

(expected) Pipe parallelepiped: 1.000000 cм^3

TEST CASE RESULT: true

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

TEST CASE 7

(input) Pipe height: 2 cм

(input) Pipe length: 40 cм

(input) Pipe width: 50 cм

(output) Pipe parallelepiped: 4000.000000 cм^3

(expected) Pipe parallelepiped: 4000.000000 cм^3

TEST CASE RESULT: true

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

TEST CASE 8

(input) Pipe height: 43.21 cм

(input) Pipe length: 81.78 cм

(input) Pipe width: 12.523 cм

(output) Pipe parallelepiped: 44252.695312 cм^3

(expected) Pipe parallelepiped: 44252.695312 cм^3

TEST CASE RESULT: true

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

TEST CASE 9

(input) Pipe height: 60 cм

(input) Pipe length: 35 cм

(input) Pipe width: 12 cм

(output) Pipe parallelepiped: 25200.000000 cм^3

(expected) Pipe parallelepiped: 25200.000000 cм^3

TEST CASE RESULT: true

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

TEST CASE 10

(input) Pipe height: 29 cм

(input) Pipe length: 143.5 cм

(input) Pipe width: 48.32 cм

(output) Pipe parallelepiped: 201083.671875 cм^3

(expected) Pipe parallelepiped: 201083.671875 cм^3

TEST CASE RESULT: true

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

# ДОДАТОК Б

**…/Teacher/ModulesNazarevskyi.h**

#ifndef MODULESNAZAREVSKYI\_H\_INCLUDED

#define MODULESNAZAREVSKYI\_H\_INCLUDED

#include <string>

#include <cstring>

#include <sstream>

#include <regex>

using namespace std;

class ClassLab12\_Nazarevskyi

{

public:

ClassLab12\_Nazarevskyi();

float getPipeHeight() { return pipeHeight; }

float getPipeLength() { return pipeLength; }

float getPipeWidth() { return pipeWidth; }

float getPipeParallelepiped() { return pipeParallelepiped(pipeHeight, pipeLength, pipeWidth); }

void setPipeHeight(string height, float&f);

void setPipeLength(string radius, float&f);

void setPipeWidth(string radius, float&f);

private:

float pipeHeight;

float pipeLength;

float pipeWidth;

float pipeParallelepiped(float height, float length, float width) { return length \* width \* height; };

};

ClassLab12\_Nazarevskyi::ClassLab12\_Nazarevskyi()

{

pipeHeight = 1;

pipeLength = 1;

pipeWidth = 1;

}

void ClassLab12\_Nazarevskyi::setPipeHeight(string height, float&f)

{

pipeHeight = 1;

char junk;

height = regex\_replace(height, regex(","), ".");

stringstream ss(height);

if (ss >> f && !(ss >> junk)) {

if (stof(height) >= 1 && stof(height) <= 60) {

pipeHeight = stof(height);

}

}

}

void ClassLab12\_Nazarevskyi::setPipeLength(string length, float&f)

{

pipeLength = 1;

char junk;

length = regex\_replace(length, regex(","), ".");

stringstream ss(length);

if (ss >> f && !(ss >> junk)) {

if (stof(length) >= 1 && stof(length) <= 150) {

pipeLength = stof(length);

}

}

}

void ClassLab12\_Nazarevskyi::setPipeWidth(string width, float&f)

{

pipeWidth = 1;

char junk;

width = regex\_replace(width, regex(","), ".");

stringstream ss(width);

if (ss >> f && !(ss >> junk)) {

if (stof(width) >= 1 && stof(width) <= 60) {

pipeWidth = stof(width);

}

}

}

#endif // MODULESNAZAREVSKYI\_H\_INCLUDED

# ДОДАТОК В

**… /Teacher/TestFunctions.h**

#ifndef TESTFUNCTIONS\_H\_INCLUDED

#define TESTFUNCTIONS\_H\_INCLUDED

#include "ModulesNazarevskyi.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\_Nazarevskyi pipe, string outArea, string expArea, int index)

{

resFile << " TEST CASE " << setw(2) << index << setw(40) << "" << endl;

resFile << " (input) Pipe height: " << setw(19) << pipe.getPipeHeight() << " cм" << endl;

resFile << " (input) Pipe length: " << setw(19) << pipe.getPipeLength() << " cм" << endl;

resFile << " (input) Pipe width: " << setw(20) << pipe.getPipeWidth() << " cм" << endl;

resFile << " (output) Pipe parallelepiped: " << setw(11) << outArea << " cм^3" << endl;

resFile << " (expected) Pipe parallelepiped: " << setw(11) << expArea << " cм^3" << endl;

resFile << " TEST CASE RESULT: " << setw(21) << boolalpha << (expArea.compare(outArea) == 0) << "" << endl;

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

}

#endif // TESTFUNCTIONS\_H\_INCLUDED

**… /Teacher/main.cpp**

#include "TestFunctions.h"

using namespace std;

int main()

{

if (fileInDirectory() == false) { return 0; }

ClassLab12\_Nazarevskyi pipe;

ifstream caseFile("../../TestSuite/TS.txt");

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

if (filesIsOpen(resultFile, caseFile) == false) { return 0; }

for (int i = 1; i < 11; i++) {

string pipeHeight;

string pipeLength;

string pipeWidth;

string pipeParallelepiped;

string bufLine;

float bufFloat;

getline(caseFile, bufLine);

pipeHeight = bufLine.erase(0, 12);

getline(caseFile, bufLine);

pipeLength = bufLine.erase(0, 12);

getline(caseFile, bufLine);

pipeWidth = bufLine.erase(0, 11);

getline(caseFile, bufLine);

pipeParallelepiped = bufLine.erase(0, 20);

pipe.setPipeHeight(pipeHeight, bufFloat);

pipe.setPipeLength(pipeLength, bufFloat);

pipe.setPipeWidth(pipeWidth, bufFloat);

bufLine = to\_string(pipe.getPipeParallelepiped());

declareTestSuiteResults(resultFile, pipe, bufLine, pipeParallelepiped, i);

}

caseFile.close();

resultFile.close();

system("pause");

return 0;

}