Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**Звіт**

Про виконання лабораторної роботи №2

**На тему:**

«Основи модульного тестування програм з використанням мови С++»

з дисципліни “Об’єктно-орієнтоване програмування”

**Лектор:**

доц. каф. ПЗ

Коротєєва Т.О.

**Виконав:**

ст. гр. ПЗ-18

Юшкевич А.І.

**Прийняв:**

асис. каф. ПЗ

Дивак І.В.

« … » … 2023 р.

∑ = \_\_\_\_\_\_\_\_\_\_\_\_

Львів – 2022

**Тема**: Основи модульного тестування програм з використанням мови С++

**Мета**: Навчитись створювати та виконувати модульні тести з використанням бібліотеки Google Test

Теоретичні відомості

**Створення простого тесту**

Щоб створити простий тест, потрібно виконати такі кроки:

1. Використайте макрос TEST() для визначення та іменування функції тесту. Функції тесту не повертають значення.
2. Всередині функції поряд із іншим C++ кодом використати різноманітні ствердження Google Test для перевірки значень.
3. Результат тесту визначається ствердженнями. Якщо хоча б одне із стверджень не успішне (фатально чи не фатально), чи якщо тест призводить до аварійного завершення, тоді увесь тест неуспішний. Інакше, він успішний.
4. Явний успіх чи неуспіх. До цієї групи належать такі макроси:
   1. FAIL() – генерує неуспішне
   2. SUCCEED() – генерує успіх
5. Перевірки булевої умови
   1. EXPECT\_TRUE(condition) та ASSERT\_TRUE(condition). Перевіряють чи умова condition істина
   2. EXPECT\_FALSE(condition) та ASSERT\_FALSE(condition). Перевіряють чи умова condition хибна
6. Бінарне порівняння
   1. EXPECT\_EQ(val1,val2) та ASSERT\_EQ(val1,val2). Перевіряють чи val1==val2.
   2. EXPECT\_NE(val1,val2) та ASSERT\_NE(val1,val2). Перевіряють чи val1!=val2.
   3. EXPECT\_LT(val1,val2) та ASSERT\_LT(val1,val2). Перевіряють val1<val2.
   4. EXPECT\_LE(val1,val2) та ASSERT\_LE(val1,val2). Перевіряють val1<=val2.
   5. EXPECT\_GT(val1,val2) та ASSERT\_GT(val1,val2). Перевіряють val1>val2.
   6. EXPECT\_GE(val1,val2) та ASSERT\_GE(val1,val2). Перевіряють val1>=val2.
7. Порівняння рядків
   1. EXPECT\_STREQ(str1,str2) та ASSERT\_STREQ(str1,str2). Перевіряють, що два C рядки (тип char \*) мають однаковий вміст.
   2. EXPECT\_STRNE(str1,str2) та ASSERT\_STRNE(str1,str2). Перевіряють, що два C рядки (тип char \*) мають різний вміст.
   3. EXPECT\_STRCASEEQ(str1,str2) та ASSERT\_STRCASEEQ(str1,str2). Перевіряють, що два C рядки (тип char \*) мають однаковий вміст ігноруючи регістр літер.
   4. EXPECT\_STRCASENE(str1,str2) та ASSERT\_STRCASENE(str1,str2). Перевіряють, що два C рядки (тип char \*) мають різний вміст ігноруючи регістр літер.
8. Порівняння чисел із плаваючою крапкою. У зв’язку із помилками заокруглення, дуже рідко щоб два числа із плаваючою крапкою співпадали точно, тому EXPECT\_EQ для таких випадків не підходить. Google Test надає макроси які порівнюють такі числа з певною точністю (використовується підхід UPL – Units in the Last Place).
   1. EXPECT\_FLOAT\_EQ(val1,val2) та ASSERT\_FLOAT\_EQ(val1,val2). Перевіряють чи два float значення val1 та val2 приблизно однакові в межах 4 ULPs.
   2. EXPECT\_DOUBLE\_EQ(val1,val2) та ASSERT\_DOUBLE\_EQ(val1,val2). Перевіряють чи два double значення val1 та val2 приблизно однакові в межах 4 ULPs.
   3. EXPECT\_NEAR(val1,val2,abs\_error) та ASSERT\_NEAR(val1,val2,abs\_error). Перевіряють чи різниця між val1 та val2 не перевищує абсолютну межу похибки abs\_error.
9. Ствердження для виключень. Дані макроси перевіряють чи частина коду генерує, або не генерує виключень.
   1. EXPECT\_THROW(statement, ex\_type) та ASSERT\_THROW(statement, ex\_type). Перевіряють чи вираз statement генерує виключення типу ex\_type
   2. EXPECT\_ANY\_THROW(statement) та ASSERT\_ANY\_THROW(statement). Перевіряють чи вираз statement генерує виключення будь-якого типу.
   3. EXPECT\_NO\_THROW(statement) та ASSERT\_NO\_THROW(statement). Перевіряють чи вираз statement не генерує жодних виключень.

Завдання

Створити тести для поданої функції:

|  |  |
| --- | --- |
| 12 | // функція повертає об'єм прямокутного паралелепіпеда  // з шириною dSideA, довжиною dSideB та висотою dHeight  // або NAN якщо вхідні дані неправильні  double GetBoxVolume(double dSideA, double dSideB, double dHeight); |

Код програми

Lab2.h:

#pragma once

#include <iostream>

using namespace std;

double GetBoxVolume(double dSideA, double dSideB, double dHeight);

Lab\_02\_OOP.cpp:

#include "lab2.h"

double GetBoxVolume(double dSideA, double dSideB, double dHeight) {

double result = NAN;

if (dSideA >= 0 && dSideB >= 0 && dHeight >= 0) {

result = dSideA \* dSideB \* dHeight;

}

return result;

}

test.cpp:

#include "pch.h"

#include "../Lab\_02\_OOP/lab2.h"

TEST(GetBoxVolume, HandlesNegativeInput) {

EXPECT\_TRUE(isnan(GetBoxVolume(8.34, 34.2, -1.34)));

EXPECT\_TRUE(isnan(GetBoxVolume(32.62, -1.37, 2.57)));

EXPECT\_TRUE(isnan(GetBoxVolume(65.32, -1.24, -1.4)));

EXPECT\_TRUE(isnan(GetBoxVolume(-1.32, 7.2, 9.34)));

EXPECT\_TRUE(isnan(GetBoxVolume(-1.32, 24.38, -1.01)));

EXPECT\_TRUE(isnan(GetBoxVolume(-1.32, -1.15, 57.3)));

EXPECT\_TRUE(isnan(GetBoxVolume(-1.32, -1.83, -1.14)));

}

TEST(GetBoxVolume, HandlesZeroInput) {

EXPECT\_DOUBLE\_EQ(GetBoxVolume(0, 0, 0), 0);

}

TEST(GetBoxVolume, HandlesCommonInput) {

EXPECT\_DOUBLE\_EQ(GetBoxVolume(8.34, 34.2, 37.34), 10'650.41352);

EXPECT\_DOUBLE\_EQ(GetBoxVolume(32.62, 0.37, 5.57), 67.226558);

EXPECT\_DOUBLE\_EQ(GetBoxVolume(65.32, 1.24, 1.4), 113.39552);

}

TEST(GetBoxVolume, HandlesLargeInput) {

EXPECT\_DOUBLE\_EQ(GetBoxVolume(1000000.0, 1000000.0, 1000000.0), 1.0e+18);

}

Результат виконання

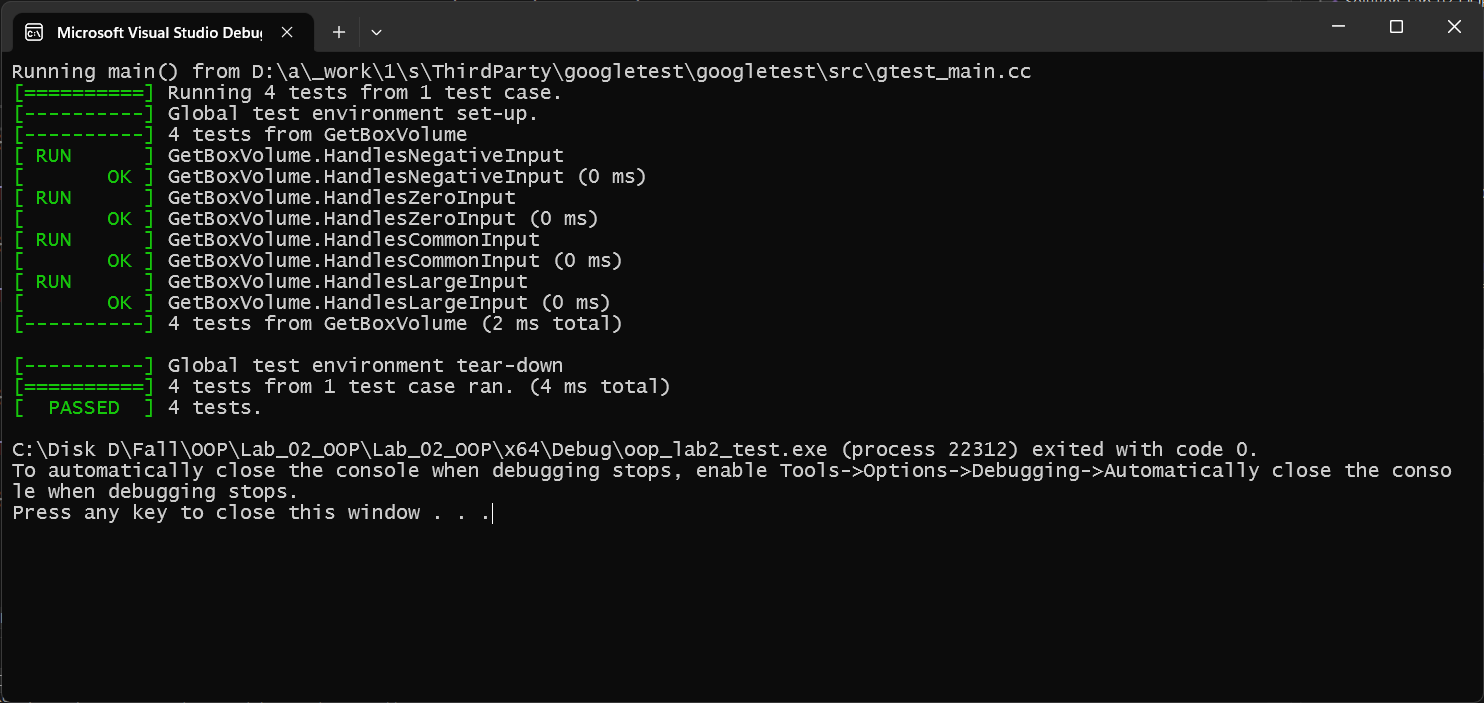


Рис 1. Виконання тестів

Висновок

Під час виконання лабораторної роботи, я навчився створювати та виконувати модульні тести з бібліотеки Google Test.