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 КБ-21

Устинович М.Є

ПЕРЕВІРИВ

ст. викладач кафедри кібербезпеки   
та програмного забезпечення

Усік П.С.

Кропивницький – 2022

**Лабораторна робота №12**

**Варіант 11**

**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.



**Аналіз вимог до програмного модуля:**

Модуль має складатися з опису класу Class Lab12\_Ustynovych, який є формальним представленням абстракції акваріуму. 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.

**Проектування програмного модуля:**

Клас ClassLab12\_Ustynovych має три значущих для даної абстракції атрибута: довжина, ширина та висота акваріума. Усі вони за вимогами мають рівень доступу private. Клас має один конструктор зі значеннями за замовчуванням замість двох констуркторів, один з яких не приймає аргументів та присвоює атрибутам значення за замовучванням, а інший – значення відповіних атрибутів. Також клас має дві функції, які отримують значення в якості аргументу та присвоюють його відповідному атрибуту, та дві функції, які повертають значення відповідного атрибуту. Останні дві функції повертають результат, що відповідає площі та периметру.

**Контрольні приклади для тестування функції надання площі та периметра:**

|  |  |  |
| --- | --- | --- |
| Test Case ID | Вхідні дані | Очікуваний результат |
| TC\_01 | width = 4; height = 5 | S = 20, p =18 |
| TC\_02 | width = 7; height = 6 | S = 42, p =26 |
| TC\_03 | width = 2; height = 9 | S = 18, p =22 |
| TC\_04 | width = 9; height = 3 | S = 27, p =24 |
| TC\_05 | width = 8; height = 4 | S = 32, p = 24 |

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 трьох вхідних значеннь та результату, розділених будь-якою кількістю відступів (пробіли, табуляції тощо). 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\_Ustynovych, 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 файла ModulesUstynovych.h наведений у Додатку А.

Лістинг вихідного коду проєкту Teacher наведено у Додатку Б.

Лістинг вихідного коду заголовкового файла Functions.h у Додатку В

Вміст файлу TestResults.txt наведений у Додатку Г.

Результати системного тестування додатка наведені у Додатку Д.

**Висновок:** дана лабораторна робота 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\_Ustynovych атрибути були захищені (описані у секції private), а методи класу, що відповідають за поведінку екземпляру класу та роботу з атрибутами – на рівні public.

Замість двох конструкторів (без параметрів та з трьома обов'язковими) було створено один конструктор, у якому реалізовані значення за замовчуванням. Клас також повинен мати метод надання об’єму акваріуму, який вираховується за формулою. Таким чином, значущими атрибутами Площі 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, перевіряючи це значення (усі атрибути є мірами довжини, тож не можуть бути меншими за 0). Метод надання об’єму повертає результат множення усіх атрибутів.

Після програмної реалізації класу у заголовковому файлі ModulesUstynovych.h, що був створений під час лабораторної роботи 12, 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, починається пошук цього під-рядка у рядку шляху. Якщо під-рядок не знайдено – відбуваються вищезазначені дії та процес повертає код -1, інакше відбувається unit-тестування.

Для unit-тестування у директорії lab12\TestSuite\ створюються текстові файли, у яких зазначені тест-кейси у вигляді вхідних даних та очікуваного результату, розділених будь-якою кількістю відступів, які для зручності зазначені на окремих рядках. Для того, щоб отримати доступ до цих файлів, потрібно мати шлях до цієї директорії. Проте для того, щоб ПЗ не залежало від окремої системи, шлях до потрібної теки виражений не повним шляхом, а відносним. Складність такого підходу в тому, що проект та додаток цього проекту, який в кінцевому результаті знаходиться в директорії lab12\Software, знаходяться на різних рівнях, тому і відносний шлях буде відрізнятися. Ця проблема була вирішена наступним чином: якщо основна директорія консольного додатка містить файл main.cpp, то шлях має вигляд ..\..\TestSuite, тобто опускається на 2 директорії вниз, інакше - ..\TestSuite, тобто опускається на одну директорію. Для перевірки наявності файлу створюється файловий потік, у якому відкривається файл з такою назвою у режимі читання, і якщо він відкрився – він закривається та шлях має перший вигляд, інакше – другий.

Наступним кроком є створення екземпляру класу ClassLab12\_Ustynovych, 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-сьюта та необов’язковий 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 функцію 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 місцях цих відступів) незалежно від кількості символів відступу (пробіл, табуляція тощо). Далі зчитане слово перетворюється на число з плаваючою комою (якщо це неможливо – на число 0), передається у функцію зміни атрибуту та потім результат функції надання атрибуту порівнюють з наступним словом, перетвореним на число з плаваючою комою. Для коректної конвертації рядку у число з плаваючою комою використовується кодування CP1251.

Результати тестування належно протоколюються 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 неможливо 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 Площі за допомогою функцій зміни атрибутів. Також через те, що функція надання об’єму перемножує значення з плаваючою комою, її значення порівнюється з очікуваним результатом з обмеженою точністю (1/1000).

**ДОДАТОК А**

**Лістинг заголовкового файлу ModulesUstynovych.h:**

#ifndef MODULESUSTYNOVYCH\_H\_INCLUDED

#define MODULESUSTYNOVYCH\_H\_INCLUDED

#include <iostream>

#include <cstring>

#include <string>

#include <iomanip>

#include <fstream>

#include <sstream>

using namespace std;

class ClassLab12\_Ustynovych

{

public:

ClassLab12\_Ustynovych(double wid, double h);

ClassLab12\_Ustynovych();

void setWidth(double wid);

void setHeight (double h);

double getWidth() {return width;}

double getHeight() {return height;}

double Perimeter() {return 2\*(width+height);}

double Square() {return width\*height;}

private:

double width, height;

};

ClassLab12\_Ustynovych::ClassLab12\_Ustynovych(double wid, double h)

{

width = wid;

height = h;

}

ClassLab12\_Ustynovych::ClassLab12\_Ustynovych()

{

width = 5.00;

height = 4.00;

}

void ClassLab12\_Ustynovych::setWidth(double wid)

{

if(wid < 0) { wid = 0;}

width = wid;

}

void ClassLab12\_Ustynovych::setHeight(double h)

{

if(h < 0) { h = 0;}

height = h;

}

#endif // MODULESUSTYNOVYCH\_H\_INCLUDED

**ДОДАТОК Б**

**Лістинг вихідного коду проекту Teacher:**

#include "ModulesUstynovych.h"

#include "Functions.h"

using namespace std;

string tsDirPath = "./";

string testResPath = "TestResults.txt";

int main()

{

system("chcp 1251 & cls");

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

if (path.find("lab12\\prj\\") == string::npos) {

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();

ClassLab12\_Ustynovych flag;

string examples = tsDirPath + "TestSuite.txt";

string results = tsDirPath + "TestResults.txt";

if (checkFileDirectory(results) == false) { return 1; }

if (checkFileOpen(results, examples) == false) { return 2; }

string buff;

ofstream resOut(results, ios::app);

if (resOut.is\_open()) {

ifstream resTest(examples);

if (resTest.is\_open()) {

bool newLine = true;

resOut << "+" << setw(34) << left << setfill('-') << "" << setw(45)

<< "+" << endl << setfill(' ');

for (int i = 0, j = 0; resTest.tellg() != -1; i++) {

if (newLine) {

resOut << "|TC\_" << left << setw(2) << j+1 << "|";

newLine = false;

}

getline(resTest, buff);

buff = buff.erase(0, 11);

cout << stod(buff) << endl;

switch (i%4) {

case 0:

resOut << "h - " << setw(8) << buff;

flag.setHeight(stod(buff));

break;

case 1:

resOut << "w - " << setw(8) << buff;

flag.setWidth(stod(buff));

break;

case 2:

resOut << "|square - " << setw(11) << flag.Square();

resOut << resetiosflags(ios::left);

cout << (int)(flag.Square()\*1000) << " | " << (int)(stod(buff)\*1000) << endl;

break;

case 3:

resOut << "|perimeter - " << flag.Perimeter() << ' ';

resOut << resetiosflags(ios::left);

cout << (int)(flag.Perimeter()\*1000) << " | " << (int)(stod(buff)\*1000) << endl;

if(((int)(flag.Perimeter()\*1000)) == (int)(stod(buff)\*1000))

{

resOut << "Passed" << endl;

}

else

resOut << "Failed" << endl;

newLine = true;

j++;

}

}

resOut << "+" << setw(80) << setfill('-') << "+" << endl << setfill(' ');

resTest.close();

}

else {

cout << "Помилка при роботі з файлом!" << examples << "." << endl;

}

resOut.close();

}

else {

cout << "Помилка при роботі з файлом!" << results << "." << endl;

}

return 0;

}

**ДОДАТОК В**

**Лістинг заголовкового файлу Functions.h:**

#ifndef FUNCTIONS\_H\_INCLUDED

#define FUNCTIONS\_H\_INCLUDED

#include <iostream>

#include <cstring>

#include <string>

#include <iomanip>

#include <fstream>

#include <sstream>

#include "ModulesUstynovych.h"

using namespace std;

bool checkFileDirectory(string writeFile)

{

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

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

if (found == -1)

{

fstream file(writeFile, ios::out);

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

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

cout << "\a";

}

file.close();

return false;

}

return true;

fstream file(writeFile, ios::out);

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

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

cout << "\a";

}

file.close();

}

bool checkFileOpen(string writeName, string readName)

{

fstream writeFile(writeName, ios::out);

fstream readFile(readName, ios::in);

if (readFile.is\_open() == false || writeFile.is\_open() == false)

{

fstream file("TestReuslts.txt", ios::out);

file << "Встановлені вимоги порядку виконання лабораторної роботи порушено" << endl;

file.close();

return false;

}

return true;

}

void TestResults (string readName, ClassLab12\_Ustynovych flag, string outSquare, string expSquare, string outPer, string expPer, int index)

{

fstream resFile(readName, ios::app);

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

resFile << "| TEST CASE " << index << endl;

resFile << "| Flag width: " << flag.getWidth() << endl;

resFile << "| Flag height: " << flag.getHeight() << endl;

resFile << "| Flag square: " << outSquare << endl;

resFile << "| Flag square(exp): " << expSquare << endl;

resFile << "| Flag perimeter: " << outPer << endl;

resFile << "| Flag perimeter(exp): " << expPer << endl;

resFile << "| TEST CASE RESULT: " << boolalpha << (expSquare.compare(outSquare) == 0 && expPer.compare(outPer) == 0 ) << " |" << endl;

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

}

#endif // FUNCTIONS\_H\_INCLUDED

**ДОДАТОК Г**

**Вміст файлу TestResults.txt:**

+----------------------------------+--------------------------------------------

|TC\_1 |h - 5.00 w - 4.00 |square - 20 |perimeter - 18 Passed

|TC\_2 |h - 6.00 w - 7.00 |square - 42 |perimeter - 26 Passed

|TC\_3 |h - 9.00 w - 2.00 |square - 18 |perimeter - 22 Passed

|TC\_4 |h - 3.00 w - 9.00 |square - 27 |perimeter - 24 Passed

|TC\_5 |h - 8.00 w - 4.00 |square - 32 |perimeter - 24 Passed

+-------------------------------------------------------------------------------+

**ДОДАТОК Д**

|  |  |
| --- | --- |
| Назва тестового набору  Test Suite Description | TS\_Teacher |
| Назва проекта / ПЗ  Name of Project / Software | Teacher |
| Рівень тестування  Level of Testing | системний / System Testing |
| Автор тест-сьюта  Test Suite Author | Устинович Максим |
| Виконавець  Implementer | Устинович Максим |

|  |  |  |  |
| --- | --- | --- | --- |
| 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) |
| TC\_01 | 1. Створити у директорії lab12\ теку tmp 2. Перемістити проект Teacher з теки prj в теку tmp 3. Скомпілювати та запустити проект | Процес завершується з кодом -1, лунає звуковий сигнал.  У теці проекту створено файл TestResults.txt  Зміст файлу:  Встановлені вимоги порядку виконання лабораторної роботи порушено! | passed |
| TC\_02 | 1. Закрити IDE Code::Blocks 2. Видалити в теці Teacher файл TestResults.txt 3. Перемістити теку Teacher в директорію lab12\ 4. Видалити теку tmp 5. Скомпілювати та запустити проект | Процес завершується з кодом -1, лунає звуковий сигнал.  У теці проекту створено файл TestResults.txt  Зміст файлу:  Встановлені вимоги порядку виконання лабораторної роботи порушено! | passed |
| TC\_03 | 1. Закрити IDE Code::Blocks 2. Видалити в теці Teacher файл TestResults.txt 3. Перемістити теку Teacher в директорію lab12\prj 4. В теці TestSuite перейменувати файл TS\_1.txt на ~TS\_1.txt 5. Скомпілювати та запустити проект | У консоль виводиться повідомлення:  Помилка при роботі з файлом ../../TestSuite/TS-1.txt.  Помилка при роботі з файлом ../../TestSuite/TS-1.txt.  Помилка при роботі з файлом ../../TestSuite/TS-1.txt.  Процес завершується з кодом 0.  У директорії lab12\TestSuite\ створено файл TestResults.txt  Зміст файлу:  +----------------------------------+--------------------------------------------  |TC\_1 |h - 5.00 w - 4.00 |square - 20 |perimeter - 18 Passed  |TC\_2 |h - 6.00 w - 7.00 |square - 42 |perimeter - 26 Passed  |TC\_3 |h - 9.00 w - 2.00 |square - 18 |perimeter - 22 Passed  |TC\_4 |h - 3.00 w - 9.00 |square - 27 |perimeter - 24 Passed  |TC\_5 |h - 8.00 w - 4.00 |square - 32 |perimeter - 24 Passed  +-------------------------------------------------------------------------------+ | passed |