ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ КАЛУЖСКОЙ ОБЛАСТИ

«КАЛУЖСКИЙ ТЕХНИКУМ ЭЛЕТРОННЫХ ПРИБОРОВ»

Отчёт по лабораторной работе

Тестирование приложения методом белого ящика

Выполнили:

Галкин А.А.

Двойнишев Р.Э.

Студент гр. ИСИП(п) 3/2

Проверил: Скопец Д.А.

Проект предоставили:

Галкин А.А.

Двойнишев Р.Э.

Калуга, 2023 г.

Цель:

Протестировать разработанные решения.

Задачи:

1. Изучить метод тестирования белого ящика.
2. Изучить написание юнит тестов.
3. Провести проверки решений.
4. Записать проведённые тесты.

**Метод тестирования белого ящика**

**Метод белого ящика –** вид тестирования при которомтестировщик имеет доступ к коду приложения.

Различают такие типы тестирования белого ящика:

1. Модульное тестирование.
2. Тестирование на утечки памяти.

**Модульное тестирование**

Модульное тестирование (блочное тестирование или юнит-тестирование) — процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы, наборы из одного или более программных модулей вместе с соответствующими управляющими данными, процедурами использования и обработки.

Главная цель модульного тестирования заключается в проверке, что каждая единица программного кода работает должным образом.

Данный вид тестирования проводится разработчиком на этапе кодирования приложения.

Модульное тестирование – это метод тестирования Белого ящика.

**Описание задания**

Разработать код, который требует ввести три значения сторон треугольника. Ограничения:  
- три числа не могут быть определены как стороны треугольника, - если хотя бы одно из них меньше или равно 0;  
- сумма двух из них меньше третьего.  
  
В результате работы кода в терминале должно отображаться: вид треугольника по сторонам, вид треугольника по углам, его площадь.

**Python**

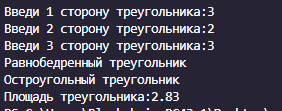


Рисунок 1 Демонстрация решения

**Решение на языке Python:**

# Проверка, что все стороны положительные

def check\_positive\_side(a,b,c):

    if a <=0 or b <=0 or c <=0:

        print("одна из сторон меньше или равна 0")

        return False

    else:

        return True

# Проверка, что сумма двух сторон больше одной стороны

def two\_side\_sum(a,b,c):

    if a+b<=c:

        print("сумма двух сторон меньше одной стороны")

    elif a+c<=b:

        print("сумма двух сторон меньше одной стороны")

    elif b+c<=a:

        print("сумма двух сторон меньше одной стороны")

    else:

        return True

    return False

# Площадь треугольника по формуле Герона

def square(a,b,c):

    p= (a+b+c)/2

    s = (p\*(p-a)\*(p-b)\*(p-c))\*\*(0.5)

    s = round(s,2)

    print(f'Площадь треугольника:{s}')

    return s

# Тип треугольника по сторонам (Равносторонний,Равнобедренный,Обычный)

def type\_triangle(a,b,c):

    type\_triangle = ""

    if a==b==c:

        type\_triangle = "Равносторонний треугольник"

    elif a==b or b==c or a==c:

        type\_triangle = "Равнобедренный треугольник"

    else:

        type\_triangle = "Обычный треугольник"

    print(type\_triangle)

    return type\_triangle

# Максимальная сторона треугольника

def max\_side( a,  b,  c):

    if (a >= b and a>=c):return a

    elif (c >= b and c>=a):return c

    else: return b

# Тип треугольника по углам (Прямоугольный,Тупоугольный,Остроугольный)

def type\_triangle\_v2(a,b,c) :

    type\_triangle = ""

    second = ""

    third =""

    max\_s = max\_side(a, b, c)

    if (a == max\_s):

        second = b

        third = c

    elif (b == max\_s):

        second = a

        third = c

    elif (c == max\_s):

        second = a

        third = b

    else: print("error")

    if (pow(max\_s, 2) == (pow(second, 2) + pow(third, 2))):

        type\_triangle = "Прямоугольный треугольник"

    elif (pow(max\_s, 2) > pow(second, 2) + pow(third, 2)):

        type\_triangle = "Тупоугольный треугольник"

    elif (pow(max\_s, 2) < pow(second, 2) + pow(third, 2)):

        type\_triangle = "Остроугольный треугольник"

    else:

        print("error")

        return

    print(type\_triangle)

    return type\_triangle

from moduleTriangle import \*

a = int(input("Введи 1 сторону треугольника:"))

b = int(input("Введи 2 сторону треугольника:"))

c = int(input("Введи 3 сторону треугольника:"))

if check\_positive\_side(a,b,c)==True and two\_side\_sum(a,b,c)==True:

    type\_triangle(a,b,c)

    type\_triangle\_v2(a,b,c)

    square(a,b,c)

**Тестирование решения на языке Python:**

assertEqual - проверяет на равенство значений действительное и ожидаемое

import unittest

from moduleTriangle import \*

class TestCalculator(unittest.TestCase):

1 тест. В ходе проведения теста ошибок не выявлено. Доработки не требует.

 def test\_positive\_side(self):

        self.assertEqual(check\_positive\_side(-10, 8, 7), False);

        self.assertEqual(check\_positive\_side(1, 2, -3), False);

        self.assertEqual(check\_positive\_side(1, 2, 0), False);

        self.assertEqual(check\_positive\_side(1, 0, -3), False);

        self.assertEqual(check\_positive\_side(6, 4, 5), True);

        self.assertEqual(check\_positive\_side(8, 7, 9), True);

        self.assertEqual(check\_positive\_side(10, 10, 11), True);

        self.assertEqual(check\_positive\_side(12, 12, 12), True);

2 тест. В ходе проведения теста ошибок не выявлено. Доработки не требует.

    def test\_two\_side\_sum(self):

        self.assertEqual(two\_side\_sum(1, 0, 2), False);

        self.assertEqual(two\_side\_sum(1, 2, 2), True);

        self.assertEqual(two\_side\_sum(1, 6, 2), False);

        self.assertEqual(two\_side\_sum(6, 2, 2), False);

        self.assertEqual(two\_side\_sum(6, 4, 5), True);

        self.assertEqual(two\_side\_sum(8, 7, 9), True);

        self.assertEqual(two\_side\_sum(10, 10, 11), True);

        self.assertEqual(two\_side\_sum(12, 12, 12), True);

3 тест. В ходе проведения теста ошибок не выявлено. Доработки не требует.

    def test\_squaer(self):

        self.assertEqual(squaer(6, 4, 5), 9.92)

        self.assertEqual(squaer(8, 9, 7), 26.83)

        self.assertEqual(squaer(10, 10, 11), 45.93)

        self.assertEqual(squaer(12, 12, 12), 62.35)

        self.assertEqual(squaer(math.exp(1), math.exp(1.2), math.exp(1.3)), 4.34);

        self.assertEqual(squaer(math.exp(2.7), 8, math.exp(2.7)), 57.33);

4 тест. В ходе проведения теста ошибок не выявлено. Доработки не требует.

 def test\_type\_triangle(self):

        self.assertEqual(type\_triangle(6, 4, 5), "Обычный треугольник");

        self.assertEqual(type\_triangle(math.exp(1), math.exp(1.2), math.exp(1.3)), "Обычный треугольник");

        self.assertEqual(type\_triangle(math.exp(2.7), 8, math.exp(2.7)), "Равнобедренный треугольник");

        self.assertEqual(type\_triangle(10, 10, 11), "Равнобедренный треугольник");

        self.assertEqual(type\_triangle(12, 12, 12), "Равносторонний треугольник");

5 тест. В ходе проведения теста ошибок не выявлено. Доработки не требует.

    def test\_max\_side(self):

        self.assertEqual(max\_side(6, 4, 5), 6);

        self.assertEqual(max\_side(6, 9, 6), 9);

        self.assertEqual(max\_side(3, 4, 5), 5);

        self.assertEqual(max\_side(12, 12, 12), 12);

        self.assertEqual(max\_side(math.exp(1), math.exp(1.2), math.exp(1.3)),math.exp(1.3) );

        self.assertEqual(max\_side(math.exp(2.3), 8, math.exp(2.7)),math.exp(2.7));

6 тест. Появляется баги при работе с float

def test\_type\_triangle\_v2(self):

self.assertEqual(type\_triangle\_v2(6, 4, 5), "Остроугольный треугольник");

        self.assertEqual(type\_triangle\_v2(6, 9, 6), "Тупоугольный треугольник");

        self.assertEqual(type\_triangle\_v2(3, 4, 5), "Прямоугольный треугольник");

        self.assertEqual(type\_triangle\_v2(5.3, 5.3, 7.5), "Прямоугольный треугольник");

        self.assertEqual(type\_triangle\_v2(6.5, 6.5, 9.19), "Прямоугольный треугольник");

        self.assertEqual(type\_triangle\_v2(12, 12, 12), "Остроугольный треугольник");

        self.assertEqual(type\_triangle\_v2(math.exp(1), math.exp(1.2), math.exp(1.3)), "Остроугольный треугольник");

        self.assertEqual(type\_triangle\_v2(math.exp(2.3), 8, math.exp(2.7)), "Тупоугольный треугольник");

if \_\_name\_\_ == "\_\_main\_\_":

  unittest.main()

**C++**

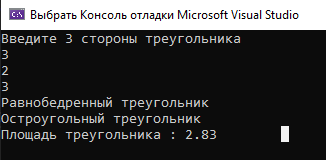


Рисунок 2 Демонстрация решения

**Решение на языке С++:**

#pragma once

#include <iostream>

#include <Windows.h>

#include <cmath>

using namespace std;

bool check\_positive\_side(float a, float b, float c);

bool two\_side\_sum(float a, float b, float c);

float square\_triangle(float a, float b, float c);

float max\_side(float a, float b, float c);

string type\_triangle(float a, float b, float c);

string type\_triangle\_v2(float a, float b, float c);

// Проверка, что все стороны положительные

bool check\_positive\_side(float a, float b, float c) {

    if (a <= 0 || b <= 0 || c <= 0) {

        cout << "одна из сторон меньше или равна 0" << endl;

        return false;

    }

    return true;

}

// Проверка, что сумма двух сторон больше одной стороны

bool two\_side\_sum(float a, float b, float c) {

    if (a + b <= c) {

        cout << "сумма двух сторон меньше одной стороны" << endl;

    }

    else if (a + c <= b) {

        cout << "сумма двух сторон меньше одной стороны" << endl;

    }

    else if (b + c <= a) {

        cout << "сумма двух сторон меньше одной стороны" << endl;

    }

    else {

        return true;

    }

    return false;

}

// Площадь треугольника по формуле Герона

float square\_triangle(float a, float b, float c) {

    float p = (a + b + c) / 2;

    float s = sqrt(p \* (p - a) \* (p - b) \* (p - c));

    s = round(s \* 100) / 100;

    cout << "Площадь треугольника : " << s << endl;

    return s;

}

//Тип треугольника по сторонам (Равносторонний,Равнобедренный,Обычный)

string type\_triangle(float a, float b, float c) {

    string type\_triagle = "";

    if (a == b && b == c && c == a) {

        type\_triagle = "Равносторонний треугольник";

    }

    else if (a == b || b == c || a == c) {

        type\_triagle = "Равнобедренный треугольник";

    }

    else {

        type\_triagle = "Обычный треугольник";

    }

    cout << type\_triagle << endl;

    return type\_triagle;

}

// Максимальная сторона треугольника

float max\_side(float a, float b, float c) {

    if (a >= b && a >= c)return a;

    else if (c >= b && c >= a)return c;

    else return b;

}

// Тип треугольника по углам (Прямоугольный,Тупоугольный,Остроугольный) через формулу Пифагора

string type\_triangle\_v2(float a, float b, float c) {

    string type\_triangle = "";

    float second;

    float third;

    float max\_s = max\_side(a, b, c);

    if (a == max\_s) {

        second = b;

        third = c;

    }

    else if (b == max\_s) {

        second = a;

        third = c;

    }

    else if (c == max\_s) {

        second = a;

        third = b;

    }

    else {

        cout << "error" << endl;

    }

        if (floor(pow(max\_s, 2) \* 10) / 10 == floor((pow(second, 2) + pow(third, 2)) \* 10) / 10) {

        type\_triangle = "Прямоугольный треугольник";

    }

    else if (pow(max\_s, 2) > pow(second, 2) + pow(third, 2)) {

        type\_triangle = "Тупоугольный треугольник";

    }

    else if (pow(max\_s, 2) < pow(second, 2) + pow(third, 2)) {

        type\_triangle = "Остроугольный треугольник";

    }

    else {

        cout << "ошибка" << endl;

    }

    cout << type\_triangle << endl;

    return type\_triangle;

}

#include "Header.h"

int main()

{

    SetConsoleCP(1251);

    SetConsoleOutputCP(1251);

    float a;

    float b;

    float c;

    cout << "Введите 3 стороны треугольника" << endl;

    cin >> a;

    cin >> b;

    cin >> c;

    if (check\_positive\_side(a, b, c) == true && two\_side\_sum(a,b,c) == true) {

       string type\_triangle1 = type\_triangle(a, b, c);

       string type\_triangle2 = type\_triangle\_v2(a, b, c);

       square(a, b, c);

    }

}

**Тестирование решения на языке С++:**

EXPECT\_EQ - проверяет на равенство значений действительное и ожидаемое (проверяет при ошибки дальше)

#include "pch.h"

#include <gtest/gtest.h>

#include <Windows.h>

#include "../../triangle/triangle/Header.h"

using namespace std;

1 тест. В ходе проведения теста ошибок не выявлено. Доработки не требует.

TEST(TestCaseName, TestPositiveSide) {

    EXPECT\_EQ(check\_positive\_side(-10, 8, 7), false);

    EXPECT\_EQ(check\_positive\_side(1, 2, 0), false);

    EXPECT\_EQ(check\_positive\_side(1, 0, -3), false);

    EXPECT\_EQ(check\_positive\_side(6, 4, 5), true);

    EXPECT\_EQ(check\_positive\_side(8, 7, 9), true);

    EXPECT\_EQ(check\_positive\_side(10, 10, 11), true);

    EXPECT\_EQ(check\_positive\_side(12, 12, 12), true);

}

2 тест. В ходе проведения теста ошибок не выявлено. Доработки не требует.

TEST(TestCaseName, TestTwo\_side\_sum) {

    EXPECT\_EQ(two\_side\_sum(1, 0, 2), false);

    EXPECT\_EQ(two\_side\_sum(1, 6, 2), false);

    EXPECT\_EQ(two\_side\_sum(6, 2, 2), false);

    EXPECT\_EQ(check\_positive\_side(6, 4, 5), true);

    EXPECT\_EQ(check\_positive\_side(8, 7, 9), true);

    EXPECT\_EQ(check\_positive\_side(10, 10, 11), true);

    EXPECT\_EQ(check\_positive\_side(12, 12, 12), true);

}

3 тест. В ходе проведения теста ошибок не выявлено. Доработки не требует.

TEST(TestCaseName, TestSquaerTriangle) {

    EXPECT\_EQ(squaer\_triangle(6, 4, 5), 9.92f);

    EXPECT\_EQ(squaer\_triangle(8, 9, 7), 26.83f);

    EXPECT\_EQ(squaer\_triangle(10, 10, 11), 45.93f);

    EXPECT\_EQ(squaer\_triangle(12, 12, 12), 62.35f);

EXPECT\_EQ(squaer\_triangle(exp(1), exp(1.2), exp(1.3)), 4.34f);

EXPECT\_EQ(squaer\_triangle(exp(2.7), 8, exp(2.7)), 57.33f);

}

4 тест. В ходе проведения теста ошибок не выявлено. Доработки не требует.

TEST(TestCaseName, TestTypeTriangle) {

    EXPECT\_EQ(type\_triangle(6, 4, 5), "Обычный треугольник");

    EXPECT\_EQ(type\_triangle(exp(1), exp(1.2), exp(1.3)), "Обычный треугольник");

EXPECT\_EQ(type\_triangle(exp(2.7), 8, exp(2.7)), "Равнобедренный треугольник");

EXPECT\_EQ(type\_triangle(10, 10, 11), "Равнобедренный треугольник");

    EXPECT\_EQ(type\_triangle(12, 12, 12), "Равносторонний треугольник");

}

5 тест. В ходе проведения теста ошибок не выявлено. Доработки не требует.

TEST(TestCaseName, TestMaxSide) {

    EXPECT\_EQ(max\_side(6, 4, 5), 6);

    EXPECT\_EQ(max\_side(6, 9, 6), 9);

    EXPECT\_EQ(max\_side(12, 12, 12), 12);

EXPECT\_EQ(max\_side(exp(1), exp(1.2), exp(1.3)), exp(1.3));

EXPECT\_EQ(max\_side(exp(2.3), 8, exp(2.7)), exp(2.7));

}

6 тест. Появляется баги при работе с float

TEST(TestCaseName, TestTypeTriangleV2) {

EXPECT\_EQ(type\_triangle\_v2(6, 4, 5), "Остроугольный треугольник");

EXPECT\_EQ(type\_triangle\_v2(6, 9, 6), "Тупоугольный треугольник");

EXPECT\_EQ(type\_triangle\_v2(5.3, 5.3, 7.49),"Прямоугольный треугольник");

EXPECT\_EQ(type\_triangle\_v2(12, 12, 12), "Остроугольный треугольник");

EXPECT\_EQ(type\_triangle\_v2(exp(1), exp(1.2), exp(1.3)), "Остроугольный треугольник");

EXPECT\_EQ(type\_triangle\_v2(exp(2.3), 8, exp(2.7)), "Тупоугольный треугольник");

}

int main(int argc, char\*\* arvg) {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

testing::InitGoogleTest(&argc, arvg);

return RUN\_ALL\_TESTS();

}

**Вывод**

Был изучен вид тестирования метод белого ящика. В ходе тестирования применён метод модульного тестирования. Для тестов была разработана программа на двух языках: пайтон и с++. Для написания тестов кода на с++ использовалась библиотека Gtest. Для кода на пайтон библиотека unittest.

В ходе тестирования были, взяты и протестированы на правильность вычислений, функции кода.

При тестировании был использован:

Охват операторов — этот метод требует, чтобы каждое возможное утверждение в коде было проверено хотя бы один раз в процессе тестирования разработки программного обеспечения.

Процент покрытия кода тестами составляет 80%. Проведённые тесты выявили ошибки с дробными числами.