Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»

Кафедра информационных систем и программирования

**ТЕСТИРОВАНИЕ И ОТЛАДКА  
ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

Отчет по лабораторной работе №4

«Тестирование, управляемое данными и анализ покрытия кода»

Выполнил:

студент 3 курса

группы 19-КБ-ПР1

Кравцов Олег Юрьевич

Проверил:

ст. преп. каф. ИСП

А. Г. Волик

Краснодар

2021

**1 Цель работы**

Изучить подход к автоматизации процесса тестирования с помощью средств среды разработки Microsoft Visual Studio, а также понятие покрытия кода (Code Coverage) тестами и научиться использовать на практике средства автоматизации определения покрытия.

**2 Задание**

1) Используя класс из задания «Лабораторной работы No2.   
Тестирование методом черноного ящика», реализующий проверку   
принадлежности точки к области и тестовые наборы из той же   
работы создать проект тестирования (или добавить еще один   
тестовый класс в уже используемый тестовый проект) с набором   
тестов управляемых данными.

2) Запустить проект тестирования и проверить результаты работы.

3) Проанализировать покрытие кода тестами.

4) На основе тестов здания «Лабораторной работы No3. Тестирование   
методом белого ящика» проанализировать покрытие кода тестами.

5) Составить отчет о результатах проведенного тестирования.

**4 Текст программы**

**4.1 Файл LR№2**

Файл Program.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace ConsoleApp5

{

class Program

{

public static void Main(string[] args)

{

Console.WriteLine("Введите X : ");

int X = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите Y : ");

int Y = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите R : ");

uint R = Convert.ToUInt16(Console.ReadLine());

if (

(X <= 0 && Y >= 0 && X >= (-1 \* R) && Y <= R && (Math.Pow(X + R, 2) + Math.Pow(Y - R, 2) >= Math.Pow(R, 2)))

||

(X >= 0 && Y <= 0 && X <= R && Y >= (-1 \* R) && (Math.Pow(X - R, 2) + Math.Pow(Y + R, 2) >= Math.Pow(R, 2)))

) Console.WriteLine("Попал!");

else Console.WriteLine("Не попал!");

}

}

public static class Point

{

public static int CheckPoint(int X, int Y, int R)

{

if (X <= 0 && Y >= 0 && X >= (-1 \* R) && Y <= R && (Math.Pow(X + R, 2) + Math.Pow(Y - R, 2) >= Math.Pow(R, 2))) return 1;

if (X >= 0 && Y <= 0 && X <= R && Y >= (-1 \* R) && (Math.Pow(X - R, 2) + Math.Pow(Y + R, 2) >= Math.Pow(R, 2))) return 2;

return 3;

}

}

}

**Текст модульных тестов**

Файл UnitTest1.cs

using NUnit.Framework;

namespace Lb\_2\_tests

{

public class Tests

{

[Test]

public void CheckLeftTopAreaTest()

{

int X = -1;

int Y = 1;

int R = 10;

int actually = ConsoleApp5.Point.CheckPoint(X, Y, R);

int expect = 1;

Assert.AreEqual(expect, actually);

}

[Test]

public void CheckRightBotAreaTest()

{

int X = 1;

int Y = -1;

int R = 10;

int actually = ConsoleApp5.Point.CheckPoint(X, Y, R);

int expect = 2;

Assert.AreEqual(expect, actually);

}

[Test]

public void CheckOuterAreaTest()

{

int X = -3;

int Y = 2;

int R = 4;

int actually = ConsoleApp5.Point.CheckPoint(X, Y, R);

int expect = 3;

Assert.AreEqual(expect, actually);

}

[Test]

public void CheckCriticalPointLeftTopAreaTest()

{

int X = -5;

int Y = 0;

int R = 5;

int actually = ConsoleApp5.Point.CheckPoint(X, Y, R);

int expect = 1;

Assert.AreEqual(expect, actually);

}

[Test]

public void CheckCriticalPointLeftTopAreaTest2()

{

int X = 0;

int Y = 5;

int R = 5;

int actually = ConsoleApp5.Point.CheckPoint(X, Y, R);

int expect = 1;

Assert.AreEqual(expect, actually);

}

[Test]

public void CheckCriticalPointRightBotAreaTest()

{

int X = 5;

int Y = 0;

int R = 5;

int actually = ConsoleApp5.Point.CheckPoint(X, Y, R);

int expect = 2;

Assert.AreEqual(expect, actually);

}

[Test]

public void CheckCriticalPointRightBotAreaTest2()

{

int X = 0;

int Y = -5;

int R = 5;

int actually = ConsoleApp5.Point.CheckPoint(X, Y, R);

int expect = 2;

Assert.AreEqual(expect, actually);

}

}

}

Файл UnitTest2.cs

using NUnit.Framework;

namespace Tests

{

class UnitTest2

{

static object[] PointsOutArea = {

new object[] { -3, 2, 4 },

new object[] { 10, 2, 3 },

new object[] { 1, -10, 6 }

};

static object[] CriticalPointsLeftTopArea = {

new object[] { -5, 0, 5 },

new object[] { 0, 5, 5 }

};

static object[] CrticalPointsRightBotArea = {

new object[] { 5, 0, 5 },

new object[] { 0, -5, 5 }

};

[TestCase(-1, 1, 10)]

[TestCase(-2, 2, 10)]

[TestCase(-5, 5, 20)]

public void CheckLeftTopAreaTest(int x, int y, int r)

{

int actually = ConsoleApp5.Point.CheckPoint(x, y, r);

int expect = 1;

Assert.AreEqual(expect, actually);

}

[TestCase(1, -1 , 10)]

[TestCase(2, -2, 10)]

[TestCase(5, -5, 20)]

public void CheckRightBotAreaTest(int x, int y, int r)

{

int actually = ConsoleApp5.Point.CheckPoint(x, y, r);

int expect = 2;

Assert.AreEqual(expect, actually);

}

[TestCaseSource("PointsOutArea")]

public void CheckOuterAreaTest(int x, int y, int r)

{

int actually = ConsoleApp5.Point.CheckPoint(x, y, r);

int expect = 3;

Assert.AreEqual(expect, actually);

}

[TestCaseSource("CriticalPointsLeftTopArea")]

public void CheckCriticalPointLeftTopAreaTest(int x, int y, int r)

{

int actually = ConsoleApp5.Point.CheckPoint(x, y, r);

int expect = 1;

Assert.AreEqual(expect, actually);

}

[TestCaseSource("CrticalPointsRightBotArea")]

public void CheckCriticalPointRightBotAreaTest(int x, int y, int r)

{

int actually = ConsoleApp5.Point.CheckPoint(x, y, r);

int expect = 2;

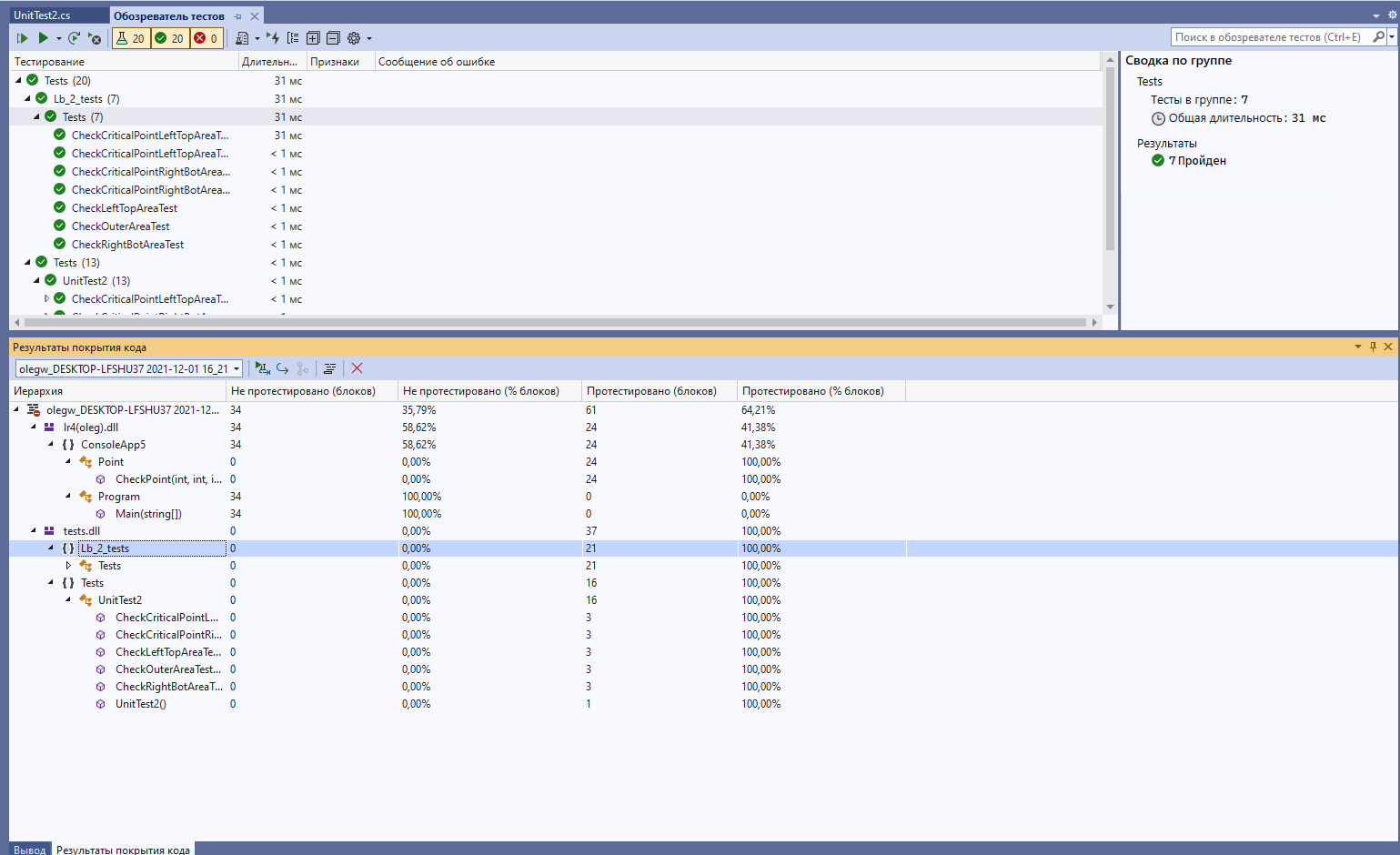
Assert.AreEqual(expect, actually);

}

}

}

**Результаты тестирования программы**



**4.2 Файл LR№3**

Файл Program.cs

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

namespace Лр\_\_\_3\_\_ТОПО\_Кравцов\_19\_КБ\_ПР1\_

{

public static class Lr\_3

{

public static void Main(string[] args)

{

}

public static string Sep(string word, string letter, string replace)

{

if (word == "")//1

return "";//2

char[] wordarr = word.ToArray();//3

for (int i = 0;//4

i < word.Length;//5

i++)//6

{

if (wordarr[i] == Convert.ToChar(letter))//7

{

wordarr[i] = Convert.ToChar(replace);//8

}

}

return new string(wordarr);//9

}

}

}

**Текст модульных тестов**

Файл StringHandlerTest.cs

using NUnit.Framework;

namespace TestProject1

{

public class Tests

{

[Test]

public void CheckEmptywordTest()

{

string word = "";

string letter = "в";

string replace = "ю";

string actually = Лр\_\_\_3\_\_ТОПО\_Кравцов\_19\_КБ\_ПР1\_.Lr\_3.Sep(word, letter, replace);

string expect = "";

Assert.AreEqual(actually, expect);

}

[Test]

public void CheckLetterTest()

{

string word = "Приветствую!";

string letter = "в";

string replace = "ю";

string actually = Лр\_\_\_3\_\_ТОПО\_Кравцов\_19\_КБ\_ПР1\_.Lr\_3.Sep(word, letter, replace);

string expect = "Приюетстюую!";

Assert.AreEqual(actually, expect);

}

[Test]

public void CheckLetterErrorTest()

{

string word = "Приветствую!";

string letter = "в";

string replace = "ю";

string actually = Лр\_\_\_3\_\_ТОПО\_Кравцов\_19\_КБ\_ПР1\_.Lr\_3.Sep(word, letter, replace);

string expect = "Приюетстюую!";

Assert.AreEqual(actually, expect);

}

[Test]

public void CheckSymbolTest()

{

string word = "Привет ствую!";

string letter = " ";

string replace = "ю";

string actually = Лр\_\_\_3\_\_ТОПО\_Кравцов\_19\_КБ\_ПР1\_.Lr\_3.Sep(word, letter, replace);

string expect = "Приветююствую!";

Assert.AreEqual(actually, expect);

}

}

}

Файл UnitTest1.cs

using NUnit.Framework;

namespace Tests

{

class UnitTest2

{

static object[] CheckSymbols = {

new object[] { "Привет ствую!", " ", "ю", "Приветююствую!" },

new object[] { "Привет!!!", "!", ".", "Привет..." },

new object[] { "Project.", ".", "/", "Project/" }

};

[TestCase("", "в", "ю", "" )]

[TestCase("", "ю", "в", "")]

[TestCase("", "o", "r", "")]

public void CheckEmptywordTest(string word, string letter, string replace, string expect)

{

string actually = Лр\_\_\_3\_\_ТОПО\_Кравцов\_19\_КБ\_ПР1\_.Lr\_3.Sep(word, letter, replace);

Assert.AreEqual(actually, expect);

}

[TestCase("Приветствую!", "в", "ю", "Приюетстюую!")]

[TestCase("Привет!", "в", "ю", "Приюет!")]

[TestCase("Project", "o", "r", "Prrject")]

public void CheckLetterTest(string word, string letter, string replace, string expect)

{

string actually = Лр\_\_\_3\_\_ТОПО\_Кравцов\_19\_КБ\_ПР1\_.Lr\_3.Sep(word, letter, replace);

Assert.AreEqual(actually, expect);

}

[TestCaseSource("CheckSymbols")]

public void CheckSymbolTest(string word, string letter, string replace, string expect)

{

string actually = Лр\_\_\_3\_\_ТОПО\_Кравцов\_19\_КБ\_ПР1\_.Lr\_3.Sep(word, letter, replace);

Assert.AreEqual(actually, expect);

}

}

}

**Результаты тестирования программы**

