МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

**ОТЧЕТ**

по лабораторной работе № 5

на тему: «Автоматизированное тестирование»

по дисциплине: «Качество и тестирование программного обеспечения»

Вариант 18

Выполнил: Шорин В.Д. Шифр: 171406

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 71ПГ

Проверили: Ужаринский А.Ю., Конюхова О.В.

Отметка о зачете:

Дата: «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Орел, 2020 г.

**Задание на лабораторную работу**

Необходимо решить задачу в соответствии с вариантом. Для решённой задачи написать модульные тесты, полностью покрывающие исходный код. Для написания модульных тестов использовать любую библиотеку модульного тестирования в зависимости от выбранного языка программирования.

**Тесты**

[TestMethod]

public void AlphabetHash\_OneChar\_OneChar()

{

string s = "a";

Dictionary<char, int> expectedHash = new Dictionary<char, int>();

for (int i = 97; i <= 122; i++)

{

expectedHash.Add((char)i, 0);

}

expectedHash[s[0]]++;

Dictionary<char, int> actualHash = Methods.AlphabetHash(s);

CollectionAssert.AreEqual(expectedHash, actualHash);

}

[TestMethod]

public void AlphabetHash\_nChar\_nChar()

{

string s = "aabbcc";

Dictionary<char, int> expectedHash = new Dictionary<char, int>();

for (int i = 97; i <= 122; i++)

{

expectedHash.Add((char)i, 0);

}

expectedHash['a'] = 2;

expectedHash['b'] = 2;

expectedHash['c'] = 2;

Dictionary<char, int> actualHash = Methods.AlphabetHash(s);

CollectionAssert.AreEqual(expectedHash, actualHash);

}

[TestMethod]

public void AreCharCountsEqual\_OneChar\_True()

{

string s = "a";

Dictionary<char, int> hash = Methods.AlphabetHash(s);

bool expected = true;

bool actual = Methods.AreCharCountsEqual(hash);

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void AreCharCountsEqual\_nCharInOrder\_True()

{

string s = "aabbccdd";

Dictionary<char, int> hash = Methods.AlphabetHash(s);

bool expected = true;

bool actual = Methods.AreCharCountsEqual(hash);

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void AreCharCountsEqual\_nCharNotInOrder\_True()

{

string s = "abcddcba";

Dictionary<char, int> hash = Methods.AlphabetHash(s);

bool expected = true;

bool actual = Methods.AreCharCountsEqual(hash);

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void AreCharCountsEqual\_nCharRandomOrder\_True()

{

string s = "anrsonoasr";

Dictionary<char, int> hash = Methods.AlphabetHash(s);

bool expected = true;

bool actual = Methods.AreCharCountsEqual(hash);

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void AreCharCountsEqual\_nCharInOrder\_False()

{

string s = "aaabccccdddd";

Dictionary<char, int> hash = Methods.AlphabetHash(s);

bool expected = false;

bool actual = Methods.AreCharCountsEqual(hash);

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void AreCharCountsEqual\_nCharNotInOrder\_False()

{

string s = "abcdddccccbba";

Dictionary<char, int> hash = Methods.AlphabetHash(s);

bool expected = false;

bool actual = Methods.AreCharCountsEqual(hash);

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void AreCharCountsEqual\_nCharRandomOrder\_False()

{

string s = "anrsonooaaaar";

Dictionary<char, int> hash = Methods.AlphabetHash(s);

bool expected = false;

bool actual = Methods.AreCharCountsEqual(hash);

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void CharCountsWithinOne\_nChar\_True()

{

string s = "aabbccc";

Dictionary<char, int> hash = Methods.AlphabetHash(s);

bool expected = true;

bool actual = Methods.CharCountsWithinOne(hash);

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void CharCountsWithinOne\_nCharRandomOrder\_True()

{

string s = "aacccbb";

Dictionary<char, int> hash = Methods.AlphabetHash(s);

bool expected = true;

bool actual = Methods.CharCountsWithinOne(hash);

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void CharCountsWithinOne\_nChar\_False()

{

string s = "aabbcccc";

Dictionary<char, int> hash = Methods.AlphabetHash(s);

bool expected = false;

bool actual = Methods.CharCountsWithinOne(hash);

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void CharCountsWithinOne\_nCharRandomOrder\_False()

{

string s = "aaccccbb";

Dictionary<char, int> hash = Methods.AlphabetHash(s);

bool expected = false;

bool actual = Methods.CharCountsWithinOne(hash);

Assert.AreEqual(expected, actual);

}

Результат работы:

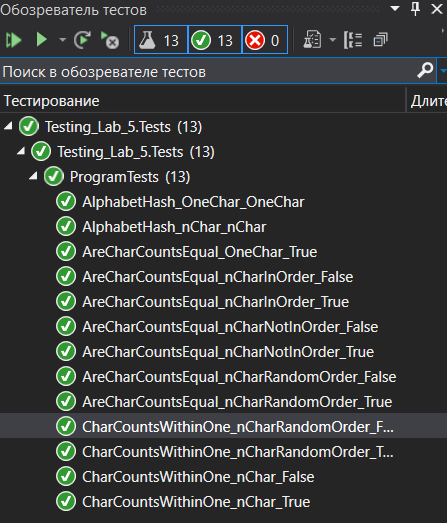


Рисунок 1 – Результат работы тестов

**Код**

**«Program.cs»**

using System;

using System.Collections.Generic;

namespace Testing\_Lab\_5

{

public static class Program

{

static void Main(string[] args)

{

string s = Console.ReadLine();

Dictionary<char, int> sHash = Methods.AlphabetHash(s);

bool answer = Methods.AreCharCountsEqual(sHash);

if (answer)

{

Console.WriteLine("YES");

}

else

{

answer = Methods.CharCountsWithinOne(sHash);

if (answer)

{

Console.WriteLine("YES");

}

else

{

Console.WriteLine("NO");

}

}

}

}

public static class Methods

{

// Считает количество букв в строке

public static Dictionary<char, int> AlphabetHash(string s)

{

Dictionary<char, int> Hash = new Dictionary<char, int>();

for (int i = 97; i <= 122; i++)

{

Hash.Add((char)i, 0);

}

for (int i = 0; i < s.Length; i++)

{

Hash[s[i]]++;

}

return Hash;

}

public static bool AreCharCountsEqual(Dictionary<char, int> Hash)

{

bool answer = false;

int countAllChars = 0;

int countCharsNumber = 0;

for (int i = 97; i <= 122; i++)

{

for (int j = 97; j <= 122; j++)

{

if (Hash[(char)i] > 0 && Hash[(char)j] > 0)

{

countAllChars++;

if (Hash[(char)i] == Hash[(char)j])

{

countCharsNumber++;

}

}

}

if (countCharsNumber == countAllChars && countCharsNumber > 0)

{

answer = true;

break;

}

else

{

countCharsNumber = 0;

countAllChars = 0;

}

}

return answer;

}

public static bool CharCountsWithinOne(Dictionary<char, int> Hash)

{

bool answer = false;

int count = 0;

int count1 = 0;

for (int i = 97; i <= 122; i++)

{

Dictionary<char, int> CheckHash = new Dictionary<char, int>(Hash);

CheckHash[(char)i]--;

for (int j = 97; j <= 122; j++)

{

for (int k = 97; k <= 122; k++)

{

if (CheckHash[(char)j] > 0 && CheckHash[(char)k] > 0)

{

if (CheckHash[(char)j] == CheckHash[(char)k])

{

count++;

}

count1++;

}

}

if (count == count1 && count > 0)

{

answer = true;

break;

}

else

{

count = 0;

count1 = 0;

}

}

if (answer)

{

break;

}

}

return answer;

}

}

}