|  |
| --- |
| ФГБОУ ВО ЮГОРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ |
| ИНЖЕНЕРНАЯ ШКОЛА ЦИФРОВЫХ ТЕХНОЛОГИЙ |

ОТЧЕТ

по лабораторной работе №1

Тема:

«Модульное тестирование»

По дисциплине:

# «Тестирование и отладка программного обеспечения»

Выполнил: студент группы

1521б

Зорин Владислав Сергеевич

Проверил: старший преподаватель

Усманов Руслан Талгатович

г. Ханты-Мансийск, 2024 г.

# Цель работы: получить навыки проведения модульного тестирования на языке программирования С#.

# Ход работы

1)Были написаны все необходимые функции (Приложение А).

2)Были написаны тесты (Приложение Б).

3)Все тесты были успешно пройдены (Рис.1).

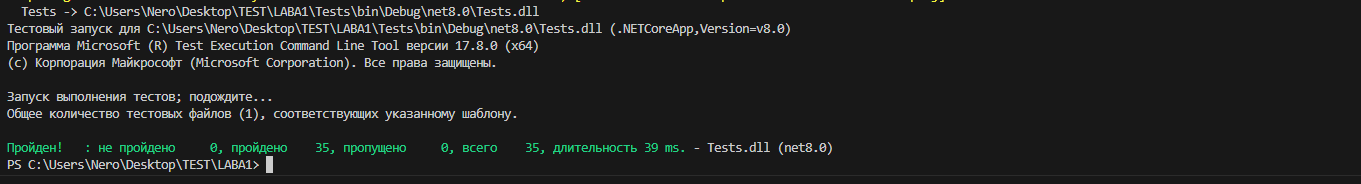


Рисунок 1 – Тестирование.

**Заключение**:

В ходе выполнения лабораторной работы я научился проводить модульное тестирование.

**Приложение А.** Листинг файла Program.cs

using System;

using System.Text;

namespace LABA

{

public class toTest

{

public int[] Sort(int[] arr)

{

if (arr.Length == 0)

{

return [-1];

}

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

{

for (int j = 0; j < arr.Length - 1 - i; j++)

{

if (arr[j] > arr[j + 1])

{

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

return arr;

}

public bool Palindrom(string s)

{

if (s == "")

{

return false;

}

for (int i = 0, j = s.Length - 1; i < j; i++, j--)

if (s[i] != s[j])

return false;

return true;

}

public int Factorial(int x)

{

if (x < 0 || x >= 13)

{

return -1;

}

int a = 1;

for (int i = x; i > 0; i--)

{

a \*= i;

}

return a;

}

public int Fibonachi(int n)

{

if (n < 0 || n > 47)

{

return -1;

}

if (n == 0 || n == 1) return n;

return Fibonachi(n - 1) + Fibonachi(n - 2);

}

public bool Includes(string s, string sub)

{

if (sub.Length > s.Length)

{

return false;

}

else if (s == "" && sub == "")

{

return true;

}

else if (s == "" || sub == "")

{

return false;

}

return s.Contains(sub);

}

public bool simpleNumber(int n)

{

if (n < 0)

{

return false;

}

for (int i = 2; i < n - 1; i++)

if (n % i == 0)

return false;

return true;

}

public int Reverse(int number)

{

string number2 = number.ToString();

if (number2.Length == 1)

{

return number;

}

string reverse = new string(number2.Reverse().ToArray());

char minus = reverse[reverse.Length - 1];

if (minus == '-')

{

string res = minus + reverse.Substring(0, reverse.Length - 1);

if (long.Parse(res) <= 2147483647 && long.Parse(res) >= -2147483647)

{

return int.Parse(res);

}

}

if (long.Parse(reverse) <= 2147483647 && long.Parse(reverse) >= -2147483647)

{

return int.Parse(reverse);

}

return 0;

}

public string toRoman(int number)

{

if (number < 0 || number > 3999)

return "error";

if (number == 0) return "N";

int[] values = new int[] { 1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1 };

string[] dict = new string[] { "M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I" };

StringBuilder result = new StringBuilder();

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

{

while (number >= values[i])

{

number -= values[i];

result.Append(dict[i]);

}

}

return result.ToString();

}

}

public class program

{

public static void Main()

{

}

}

}

**Приложение Б.** Листинг файла UnitTest1.cs

using LABA;

using NUnit.Framework.Interfaces;

namespace Tests

{

public class Tests

{

private toTest obj;

[SetUp]

public void Setup()

{

obj = new toTest();

}

[TestCase(new int[] { 1, 0, 3, 4 }, new int[] { 0, 1, 3, 4 })]

[TestCase(new int[] { -1, 0, -3, 4 }, new int[] { -3, -1, 0, 4 })]

[TestCase(new int[] { 1, 0, 4, 2 }, new int[] { 0, 1, 2, 4 })]

[TestCase(new int[] { }, new int[] { -1 })]

public void Sort(int[] n, int[] res)

{

Assert.AreEqual(res, obj.Sort(n));

}

[TestCase("1111", true)]

[TestCase("abab", false)]

[TestCase("a", true)]

[TestCase("", false)]

public void Palindrom(string n, bool res)

{

Assert.AreEqual(res, obj.Palindrom(n));

}

[TestCase(-123, -1)]

[TestCase(100, -1)]

[TestCase(5, 120)]

[TestCase(6, 720)]

public void Factorial(int n, int res)

{

Assert.AreEqual(res, obj.Factorial(n));

}

[TestCase(0, 0)]

[TestCase(-123, -1)]

[TestCase(123, -1)]

[TestCase(3, 2)]

public void Fibonachi(int n, int res)

{

Assert.AreEqual(res, obj.Fibonachi(n));

}

[TestCase("", "", true)]

[TestCase("abcdefg", "a", true)]

[TestCase("", "c", false)]

[TestCase("abcdefg", "", false)]

public void Includes(string s, string sub, bool res)

{

Assert.AreEqual(res, obj.Includes(s, sub));

}

[TestCase(-1, false)]

[TestCase(4, false)]

[TestCase(5, true)]

[TestCase(64, false)]

public void simpleNumber(int n, bool res)

{

Assert.AreEqual(res, obj.simpleNumber(n));

}

[TestCase(-12, -21)]

[TestCase(-1, -1)]

[TestCase(200, 2)]

[TestCase(2147483647, 0)]

[TestCase(0, 0)]

public void Reverse(int n, int res)

{

Assert.AreEqual(res, obj.Reverse(n));

}

[TestCase(1, "I")]

[TestCase(100, "C")]

[TestCase(1010, "MX")]

[TestCase(0, "N")]

[TestCase(-1, "error")]

[TestCase(4000, "error")]

public void toRoman(int n, string res)

{

Assert.AreEqual(res, obj.toRoman(n));

}

}

}