|  |
| --- |
| 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

Алексеев Кирилл Евгеньевич

Проверил: старший преподаватель

Усманов Руслан Талгатович

г. Ханты-Мансийск, 2023 г.

**Цель работы:** Получение практических навыков по созданию модульных тестов.

# Ход работы

На языке C# был написан статический класс toTest, который содержит следующие методы (см. Приложение А):

1.1. Функция, выполняющая сортировку (любым алгоритмом на выбор) массива целых чисел.

1.2. Функция, которая проверяет является ли введенная строка палиндромом.

1.3. Функция, выполняющая вычисление факториала для целого числа.

1.4. Функция, которая возвращает число Фибоначчи на указанной позиции.

1.5. Функция, выполняющая поиск подстроки в строке.

1.6. Функция, проверяющая является ли число простым.

1.7. Функция, которая принимает целое 32-битное число х, и возвращает это число с цифрами в обратном порядке. Например, для 123 возвращается 321; для -120 возвращается -21. Если число х после инверсии превышает размер 32 бит, то тогда следует вернуть 0.

1.8. Функция, которая принимает целое число х и возвращает это число в римской системе счисления.

Затем были написаны Unit-тесты на Фреймворке NUnit (см. Приложение Б). Результат тестирования показан на рисунке 1.

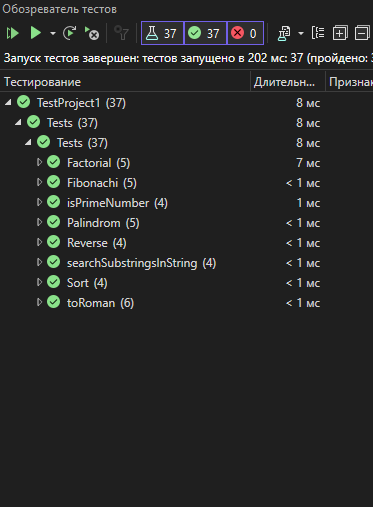


Рисунок 1 – итоги теста методов класса toTest.

**Заключение**:

В ходе выполнения лабораторной работы были написаны Unit-тесты.

**Приложение А.** Листинг файла UnitTest1.cs

using ConsoleApp1;

namespace Tests

{

public class Tests

{

[SetUp]

public void Setup()

{

}

//1.1

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

[TestCase(new int[] { 5, 2, 7, 1, 3 }, new int[] { 1, 2, 3, 5, 7 })]

[TestCase(new int[] { 9, 8, 7, 6, 5 }, new int[] { 5, 6, 7, 8, 9 })]

[TestCase(new int[] { 4, 3, 2, 1 }, new int[] { 1, 2, 3, 4 })]

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

{

toTest.Sort(ref n);

Assert.AreEqual(res, n);

}

//1.2

[TestCase("1111", true)]

[TestCase("radar", true)]

[TestCase("level", true)]

[TestCase("hello", false)]

[TestCase("amanaplanacanalpanama", true)]

public void Palindrom(string n, bool res)

{

Assert.AreEqual(res, toTest.isPalindrome(n));

}

//1.3

[TestCase(3, 6)]

[TestCase(0, 1)]

[TestCase(5, 120)]

[TestCase(7, 5040)]

[TestCase(10, 3628800)]

public void Factorial(int n, int res)

{

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

}

//1.4

[TestCase(1, 0)]

[TestCase(2, 1)]

[TestCase(5, 3)]

[TestCase(8, 13)]

[TestCase(10, 34)]

public void Fibonachi(int n, int res)

{

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

}

//1.5

[TestCase("abcdefg", "a", 0)]

[TestCase("abcdefg", "b", 1)]

[TestCase("abcdefg", "g", 6)]

[TestCase("abcdefg", "q", -1)]

public void searchSubstringsInString(string s, string sub, int res)

{

Assert.AreEqual(res, toTest.searchSubstringsInString(s, sub));

}

//1.6

[TestCase(1, true)]

[TestCase(4, false)]

[TestCase(5, true)]

[TestCase(64, false)]

public void isPrimeNumber(int n, bool res)

{

Assert.AreEqual(res, toTest.isPrimeNumber(n));

}

//1.7

[TestCase(-12, -21)]

[TestCase(2147483647, 0)]

[TestCase(-10, -1)]

[TestCase(200, 2)]

public void Reverse(int n, int res)

{

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

}

//1.8

[TestCase(1, "I")]

[TestCase(0, "Null")]

[TestCase(-10, "error")]

[TestCase(100, "C")]

[TestCase(1010, "MX")]

[TestCase(3, "III")]

public void toRoman(int n, string res)

{

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

}

}

}

**Приложение Б.** Листинг файла Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

static public class toTest

{

static private void qSort(ref int[] arr, int startIndex, int endIndex)

{

int l = startIndex, r = endIndex;

int piv = arr[(l + r) / 2];

while (l <= r)

{

while (arr[l] < piv)

{

l++;

}

while (arr[r] > piv)

{

r--;

}

if (l <= r)

{

(arr[l], arr[r]) = (arr[r], arr[l]);

l++;

r--;

}

}

if (startIndex < r)

{

qSort(ref arr, startIndex, r);

}

if (l < endIndex)

{

qSort(ref arr, l, endIndex);

}

}

//1.1

static public void Sort(ref int[] arr)

{

qSort(ref arr, 0, arr.Length - 1);

}

//1.2

static public bool isPalindrome(string str)

{

for (int startIndex = 0, endIndex = str.Length - 1; startIndex < endIndex; startIndex++, endIndex--)

{

if (str[startIndex] != str[endIndex])

{

return false;

}

}

return true;

}

//1.3

static public int Factorial(int x)

{

int a = 1;

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

{

a \*= i;

}

return a;

}

//1.4

static public int Fibonachi(int indexFibonacci)

{

indexFibonacci--;

if (indexFibonacci < 0)

{

return -1;

}

int[] arrFibonacci = new int[indexFibonacci + 2];

arrFibonacci[0] = 0;

arrFibonacci[1] = 1;

for (int i = 2; i < arrFibonacci.Length; i++)

{

arrFibonacci[i] = arrFibonacci[i - 1] + arrFibonacci[i - 2];

}

return arrFibonacci[indexFibonacci];

}

//1.5

static public int searchSubstringsInString(string str, string subStr)

{

int IndexSubStr = 0;

for (int indexStr = 0; indexStr < str.Length; indexStr++)

{

if (subStr[IndexSubStr] == str[indexStr])

{

IndexSubStr++;

if (IndexSubStr == subStr.Length)

{

return indexStr - IndexSubStr + 1;

}

}

else

{

IndexSubStr = 0;

}

}

return -1;

}

//1.6

static public bool isPrimeNumber(int num)

{

for (int i = 2; i <= Math.Sqrt(num); i += 2)

{

if (num % i == 0)

{

return false;

}

}

return true;

}

//1.7

static public int Reverse(int num)

{

int reversed\_num = 0;

while (num != 0)

{

int digit = num % 10;

if (reversed\_num > Int32.MaxValue / 10 || (reversed\_num == Int32.MaxValue / 10 && digit > 7))

{

return 0;

}

if (reversed\_num < Int32.MinValue / 10 || (reversed\_num == Int32.MinValue / 10 && digit < -8))

{

return 0;

}

reversed\_num = reversed\_num \* 10 + digit;

num /= 10;

}

return reversed\_num;

}

//1.8

static public string toRoman(int number)

{

if (number < 0 || number > 3999)

{

return "error";

}

if (number == 0)

{

return "Null";

}

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()

{

}

}

}