Национальный Исследовательский Университет
«Московский Энергетический Институт»
Кафедра прикладной математики и искусственного интеллекта

Тема: Разработка модульного теста для кода на языке C++ с применением Microsoft C++ Unit Test Framework.

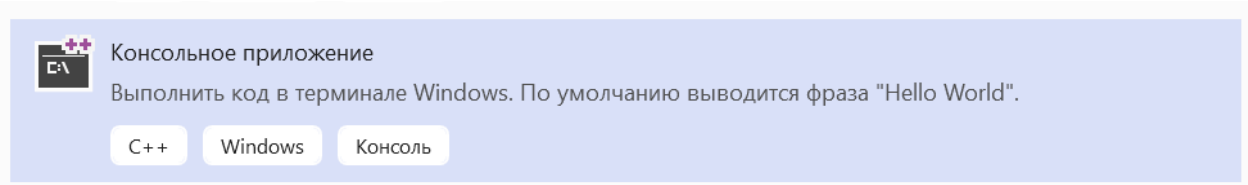
Студент: Ростовых Александра

Москва 2021

Цель работы

Научиться разрабатывать модульные тесты для кода, написанного на языке C++ с применением C++ Unit Test Framework.

1. Создать с помощью Microsoft Visual Studio 2013 консольный проект C++ (Console Application).



2 .Создать несколько классов, которые будут имитировать тестируемую логику.

1ый класс с двумя конструкторами и логическими функциями, в которых результат зависит от знака передаваемого в конструктор числа и флага, устанавливаемого *#define.*

```
#include <iostream>
#define FLAG 0
#include <cmath>

class Test1
{
public:
	double num;
	Test1() {
		num = 0;
	}
	Test1(double value) {
		num = value;
	}

	bool IsOK() {
		if (!FLAG)
		{
			if (num >= 0)
				return true;
			return false;
		}
		else
		{
			if (num >= 0)
				return false;
			return true;
		}
	}
```

```
	bool IsOK(double value) {
		num = value;
		if (!FLAG)
		{
			if (num >= 0)
				return true;
			return false;
		}
		else
		{
			if (num >= 0)
				return false;
			return true;
		}
	}
};
```

Второй класс, в котором так же несколько конструкторов, есть функция получения фамилии, функция поиска максимума и просто функция для установки (если поставлен флаг) пустого указателя и выполняющая еще некоторые действия

```
#include <string>
#include <cmath>
#include "Test1.h"
using namespace std;

class Test2
{
public:
	double num;
	string name;
	string ss;
	string* point;

	Test2(string str)
	{
		if (FLAG)
			str = "";
		name = str;
	}
	Test2(double value) {
		num = value;
		if (!FLAG)
			point = NULL;
	}
	Test2() {
```

```
            num = 0;
            point = &ss;
        }
        string GetSurname() {
            return name;
        }

        int Max(int a, int b, int c)
        {
            if (num != 0)
                a = num;
            int max = a;
            if (max < b) max = b;
            if (max < c) max = c;
            if (!FLAG)
                return max;
            else
                return -max;
        }
        double SomeTest()
        {
        if (FLAG)
            point = NULL;
            if (Test1(num).IsOK())
                return num*cos(num);
            else
                return NAN;
        }

};
```

3. Разработать чек-лист (Список проверок).

- Проверка работы функции IsOK (для всех перегрузок).
- Проверка правильности работы функции поиска максимума.
- Проверка работы функции GetSurname.
- Проверка неизменности указателя после выполнения (при условии, что они и не должны меняться)

4. Создать в проекте новый тестовый класс.

Создано.

5. Разработать не менее пяти тестирующих функций. При разработке этих функций следует активно применять методы статического класса Assert.

```
using namespace Microsoft::VisualStudio::CppUnitTestFramework;
```

```
namespace UnitTest1
{
	TEST_CLASS(UnitTest1)
	{
	public:

		TEST_METHOD(TestMethod1)
		{
			Test1 Pos(0);
			Assert::AreEqual(true, Pos.IsOK());
			Assert::AreNotEqual(true, Pos.IsOK(-2));
			Assert::AreNotEqual(false, Pos.IsOK(10));
		}
		TEST_METHOD(TestMethod2)
		{
			int i = 172;
			int j = -38.5;
			Test1 Pos;
			Assert::IsTrue(Pos.IsOK(1));
			Assert::IsFalse(Pos.IsOK(-5));
			Assert::IsTrue(Pos.IsOK(log(i+j)));
		}

		TEST_METHOD(TestMethod3)
		{
			Test2 T;
			if (!FLAG)
				Assert::IsNotNull(&T.ss);
			else
				Assert::IsNull(&T.ss);
		}

		TEST_METHOD(TestMethod4)
		{
			Test2 T;
			string str = "Лабораторная №1";
			T.ss = str;
			T.SomeTest();
			Assert::AreSame(T.ss, *T.point);
		}

		TEST_METHOD(TestMethod5)
		{
			string name = "Ростовых";
			Test2 T(name);
			Assert::AreEqual(name, T.GetSurname());
		}
		TEST_METHOD(TestMethod6)
		{
			Test2 T;
```

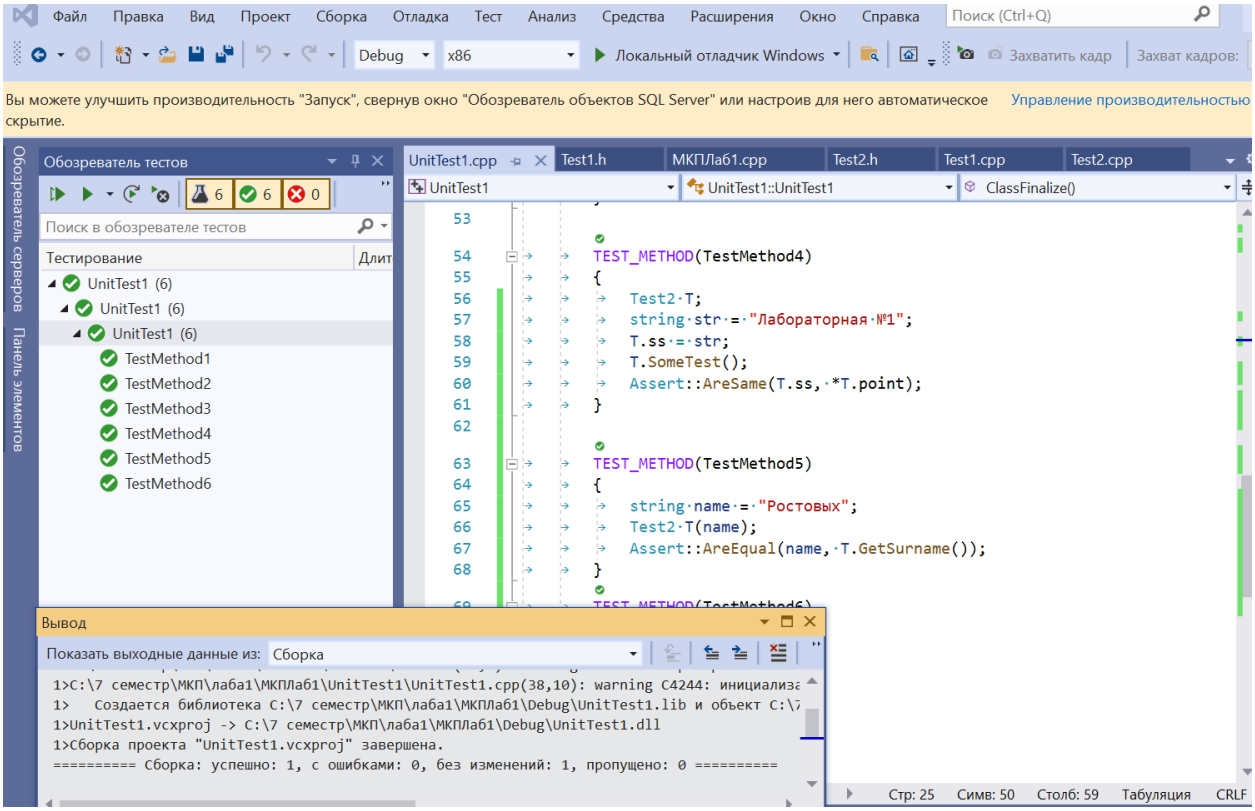
```
                int t;
                if (!FLAG)
                        t = 15;
                else
                        t = 10;
                Test2 T1(t);
                Assert::AreEqual(10, T.Max(3,10,7));
                Assert::AreNotEqual(10, T1.Max(3, 10, 7));
        }
    };
}
```

6. Реализовать функции TEST_CLASS_INITIALIZE и TEST_CLASS_CLEANUP.

```
TEST_CLASS_INITIALIZE(ClassInitialize)
        {
                Logger::WriteMessage("The test class has been created.");
        }

TEST_CLASS_CLEANUP(ClassFinalize)
        {
                Logger::WriteMessage("The test class has been cleaned.");
        }
```

7. Скомпилировать проект. С помощью меню в Test Explorer запустить тесты. Посмотреть на результат теста.



8. Внести в тестируемые классы изменения, приводящие к ошибкам.

Меняем FLAG с 0 на 1

9. Скомпилировать проект. С помощью меню в Test Explorer запустить тесты. Посмотреть, пойманы ли ошибки модульным тестом.